ERKLÄRUNG

BASIC REFERENCE CARDS

Dies ist die Sammlung der REFERENCE CARDS für die Programmiersprache ATARI–BASIC.
Wie bei einer Speisekarte befinden sich auf der Vorderseite der BASIC REFERENCE CARDS die verschiedenen Anwändungsfälle, von denen der gewünschte ausgewählt werden kann.
Stichwortartig sind in den einzelnen Zeilen die verschiedenen Spielarten des Befehls aufgeführt und beschrieben. Dabei wurde auf eine klare optische Gliederung Wert gelegt, um die Übersicht zu erhöhen.

Auf der Rückseite der REFERENCE CARDS ist, soweit sinnvoll, für jeden Befehlsteil ein Programmierrezept angegeben. Dabei stehen in der Regel die BASIC-Programmteile auf der linken Seite, die Kommentare dazu auf der rechten Seite.

Es muß darauf geachtet werden, daß das ATARI-BASIC nur Großbuchstaben versteht. Kleinbuchstaben sind nur in Anführungszeichen oder in Stringvariablen zulässig. Werden Groß- und Kleinbuchstaben wie in den Beispielen angegeben, eingetippt, gibt es keine Schwierigkeiten. Damit dabei die Sondertasten leicht erkannt werden koennen, wurden sie im Druckbild durch eine Umrahmung gekennzeichnet.

ERKLÄRUNG

Die REFERENCE CARDS sollen kein Lehrbuch sein, mit dem man das Programmieren lernen kann.
Sie stellen vielmehr ein Nachschlagewerk dar, mit dem zu einem bestimmten Befehl oder Stichwort möglichst schnell alle verfügbaren Informationen auf einen Blick zur Verfügung gestellt werden sollen. Dabei wird jeder Befehlsteil durch ein Programmrezept in ATARI-BASIC nachvollziehbar.

Querverweise auf andere Befehle und Stichworte mit ähnlicher oder entgegengesetzter Anwendung stellen wertvolle Hinweise dar, um das Gesamtbild abzurunden.

INHALTSVERZEICHNIS

GRAPHICS – PM-GRAFIK 3

INHALTSVERZEICHNIS

INHALTSVERZEICHNIS

STICHWORTVERZEICHNIS

STICHWORTVERZEICHNIS

Stichwort	**auf Karte**
Cosinus einer Zahl	COS
Cursor	
– Text	Tasten Cursor, PRINT Taste [ESC]
– Grafik	POSITION

STICHWORTVERZEICHNIS

STICHWORTVERZEICHNIS

F

STICHWORTVERZEICHNIS



STICHWORTVERZEICHNIS

S – SPEICHERN

SPRUNG – STRING


V – W

BASIC Befehlsworte

BASIC-Worte	zu finden unter
ENTER"D : PROGNAME.ERW" / ENTER"C : "	ENTER
EXP(X)	EXP
FOR X=Y TO Z STEP R: . . . :NEXT X	FOR/NEXT
FRE(0)	FRE
GET #N , A	GET
GOSUB Zeilennummer / GOSUB A	GOSUB/RETURN
GOTO Zeilennummer / GOTO A	GOTO
GRAPHICS N	GRAPHICS
IF . . . THEN	IF/THEN
INPUT A , B$ / INPUT #N , A$	INPUT
INT (X)	INT
LEN (A$) / LEN ("TEXT")	LEN
LET ENDE=Z	LET
LIST / LIST Zeilennummer , Zeilennummer	LIST
LIST"D : PROGNAME.ERW" / ,Zeilennummer , Zeilennummer	LIST
LOAD"D : PROGNAME.ERW" / LOAD"C : "	LOAD
LOCATE X , Y , A	LOCATE
LOG (X)	LOG
LPRINT "HALLO" ; "TEST" , A$, CHR$(A)	LPRINT
NEW	NEW
NEXT X	FOR/NEXT

BASIC Befehlsworte

BASIC-Worte	zu finden unter
(NOT A)	OPERATOREN
NOTE #N , X , Y	NOTE
ON . . . GOSUB Zeilennummer , Zeilennummer	ON/GOSUB/RETURN
ON . . . GOTO Zeilennummer , Zeilennummer	ON/GOTO
OPEN #N , A , B , "D : PROGNAME.ERW"	OPEN/CLOSE
OR: IF X OR Y THEN . . .	OPERATOREN
PADDLE (N)	PADDLE
PEEK (Zellennummer)	PEEK
PLOT X , Y	PLOT
POINT #N , X , Y	POINT
POKE Zellennummer , X	POKE
POP	POP
POSITION X , Y	POSITION
PRINT "HALLO" ; TEST" , A$, CHR$(A)	PRINT
PRINT #N , A$	PRINT
PTRIG (N)	PTRIG
PUT #N , A	PUT
RAD	RAD
READ Zahl , Text	READ/DATA
REM Kommentar	REM
RESTORE Zeilennummer / RESTORE A	RESTORE
RETURN	RETURN

BASIC Befehlsworte

BASIC-Worte	zu finden unter
RND (0)	RND
RUN / RUN"D : PROGNAME.ERW" / RUNA$	RUN
SAVE"D : PROGNAME.ERW" / SAVE"C : "	SAVE
SETCOLOR A , B , C	SETCOLOR
SGN (X)	SGN
SIN (X)	SIN
SOUND N , A , B , C	SOUND
SQR (X)	SQR
STATUS #N , A	STATUS
STEP: FOR X=Y TO Z STEP R	FOR/NEXT
STICK (N)	STICK
STRIG (N)	STRIG
STOP	STOP
STR$ (Zahl)	STR$
THEN: IF A THEN . . .	IF/THEN
TO: FOR X=Y TO Z	FOR/NEXT
TRAP Zeilennummer	TRAP
Q=USR(Zellennummer) / Q=USR(ADR(A$))	USR
VAL (A$)	VAL
XIO Z , #N , A , B , "D : PROGNAME.ERW"	XIO

ERKLÄRUNG

ABS

Zum Feststellen des absoluten Wertes.

Die Funktion entfernt das Minuszeichen

1. der Zahl,
2. der Variablen oder
3. des zu berechnenden Wertes

   die/der in der Klammer hinter **ABS** steht.

1.	100	PRINT ABS(−17)	17
2.	100	A = −27 : PRINT ABS(A)	27
3.	100	PRINT ABS(10−17)	7

ERKLÄRUNG

ADR

Zum Feststellen des Beginns einer Stringvariablen.

Die Funktion

1. ermittelt die Lage der Strigvariablen im RAM-Speicher, deren Name in der Klammer hinter **ADR** steht,

2. gibt die Adresse der Speicherstelle als Dezimalzahl an,

3. wird im Zusammenhang mit **USR** verwendet, wenn ein Maschinenprogramm als Stringvariable abgelegt worden ist (siehe Befehl **USR**),

4. kann auch mit Trick zur Feststellung der Adresse einer indizierten Zahlenvariablen benutzt werden.

1.
```
100 DIM A$(7)
200 A$="OTTOKAR"
```
2.
```
300 PRINT ADR(A$)
```
2108 (der String "OTTOKAR" begint bei Zelle 2108)

3.
```
Q=USR(ADR(A$))
```
siehe Befehl USR (Vorsicht! Dieser Befehl führt zum "Absturz" des Computers wenn kein **RTS** Befehl den Computer zum BASIC zurück schickt)
Taste [RETURN] oder Computer neu einschalten bringt Abhilfe

4.
```
100 DIM A$(1) , A(2)
200 FOR X=0 TO 2
300 INPUT Z : A(X)=Z
400 NEXT X
500 AD=ADR(A$)
600 ANFANG=AD+1
700 PRINT ANFANG
```
Da die Adresse der Zeichenvariablen A$ bestimmt werden kann und A(2) direkt danach dimensioniert wurde ergibt sich die Adresse von A(2) aus ADR(A$)+1 (da A$ nur eine Zelle reserviert.)

ERKLÄRUNG

Zum Feststellen des ATASCII-Codes eines Zeichens.

Die Funktion

1. ergibt die ATASCII-Codezahl des ersten Zeichens
   a. der String-Variablen oder
   b. des Textes in Anführungszeichen,
   die/der in der Klammer hinter **ASC** steht,

2. ergibt eine Dezimalzahl zwischen 0 und 255,

3. kann beim Rechnen wie eine Zahl behandelt werden.

BEISPIELE

ASC

1a.	100 DIM A$(5) 200 A$="ANTON" 300 PRINT ASC(A$)	 65 (entspricht dem großen A)
2b.	100 A=ASC("A") 200 PRINT ASC(A)	 65
3a.	PRINT ASC("B")	66
3b.	PRINT 10+ASC("B")	76

ATASCII-Code

ATARI-Version des ASCII-Fernschreibcodes

Der ATASCII-Code

1. legt für jedes Zeichen des Zeichensatzes eine Codezahl fest, die zwischen 0 und 255 liegt, dabei haben z.B.
   a. invertierte Zeichen eine um 128 höhere Codezahl als nicht invertierte Zeichen,
   b. kleine Buchstaben eine um 32 höhere Codezahl als Große,
   c. Ziffern kleinere Codezahlen als Buchstaben,

2. ist durch seinen sinnvollen Aufbau für die Benutzung mit den Basic-Befehlen für die Stringverarbeitung besonders geeignet, und ermöglicht es dadurch leicht, alphabetische Sortierungen von Strings vorzunehmen,

3. enthält den ASCII-Code ("Amerikanischer Fernschreibcode"), auch "ISO-7 Bit-Code"), der wegen seiner Leistungsfähigkeit von vielen Mikrocomputern für die Darstellung von Steuerzeichen und alphanumerischen Zeichen verwendet wird,

4. entspricht dem ASCII-Code bei allen nicht invertierten Alpha-Numerischen Zeichen; die Codes der Steuerzeichen des ASCII-Codes werden als Grafiksymbole ausgegeben,

5. eines Zeichens kann festgestellt werden durch den Befehl **ASC**,

6. wird ausgeführt, wenn er mit dem Befehl **PRINT** oder **PUT CHR$** an den Schirmeditor oder den Drucker gesandt wird (siehe **CHR$**).

ATASCII-Code

Der Code ist in dezimaler Schreibweise angegeben.

Zei.	Code	Zei.	Code	Zei.	Code	Zei.	Code	Zei.	Code	Zei.	Code
Zwr	32	0	48	@	64	P	80	Karo	96	p	112
!	33	1	49	A	65	Q	81	a	97	q	113
"	34	2	50	B	66	R	82	b	98	r	114
#	35	3	51	C	67	S	83	c	99	s	115
$	36	4	52	D	68	T	84	d	100	t	116
%	37	5	53	E	69	U	85	e	101	u	117
&	38	6	54	F	70	V	86	f	102	v	118
'	39	7	55	G	71	W	87	g	103	w	119
(	40	8	56	H	72	X	88	h	104	x	120
)	41	9	57	I	73	Y	89	i	105	y	121
*	42	:	58	J	74	Z	90	j	106	z	122
+	43	;	59	K	75	[	91	k	107	Pik	123
,	44	<	60	L	76	\	92	l	108	|	124
–	45	=	61	M	77	]	93	m	109	CLEAR	125
.	46	>	62	N	78	^	94	n	110	DELETE	126
/	47	?	63	O	79	_	95	o	111	TAB	127
Herz	0	Kreuz	16	EOL	155	Summer	253				

(siehe Pseudo-Grafik für die Codezahlen von 0 bis 31)

ERKLÄRUNG

ATN

Zum Errechnen des Arcustangens.

Die Funktion

1. errechnet den Arcustangens
   a. der Zahl,
   b. der Variablen oder
   c. des zu berechnenden Wertes

die/der in der Klammer hinter **ATN** steht,

2. kann den Winkel im Bogenmaß oder in Grad errechnen.

(siehe Befehle **DEG** und **RAD**).

1a.	100	DEG	Umschaltung auf Grad
	200	PRINT SIN(45)/COS(45)	1 Tangens von 45°
	300	PRINT ATN(1)	45.00000033 (Rundungsfehler, eigentlich 45)
1b.	100	A=1 : PRINT ATN(A)	45.00000033
1c.	100	PRINT ATN(100−99)	45.00000033
1d.	100	PRINT ATN(0)	0
1e.	100	PRINT ATN(1E30)	89.99999971 (Rundungsfehler, eigentlich 90)
2.	100	RAD	Umschaltung auf Bogenmaß
	200	PRINT ATN(1)	0.7853981684

Hinweis:
Rundungsroutinen siehe Befehl **INT**

BASIC Interpreter

Ein Programm zum Übersetzen eines in BASIC geschriebenen Programmes in die für den Computer verständliche Maschinensprache.

Der ATARI BASIC Interpreter

1. übersetzt (Interpreter = Übersetzer) die eingegebenen BASIC-Statements (das "Quellenprogramm") für den Computer,

2. arbeitet interaktiv; d.h., daß jede Eingabe, die vom Benutzer gemacht wird, durch den Computer beantwortet wird und daß jederzeit zwischen den Betriebsarten Direkteingabe, Programmeingabe und Programmdurchführung gewechselt werden kann,

3. befindet sich in der Programmkassette mit der Aufschrift "**BASIC COMPUTER LANGUAGE**", die in den linken Programmschacht eingesteckt sein muß,

4. legt die BASIC-Programme im Speicher des Computers in einem Zwischencode ab ("tokenized Version"),

BASIC Interpreter

5. übersetzt bei der Programmdurchführung den Zwischencode jedesmal neu in die Maschinensprache des benutzten Prozessors,

6. ist deshalb 10 bis 1000 mal langsamer als ein entsprechendes Maschinenprogramm,

7. arbeitet bei mathematischen Aufgaben intern im BCD-Code und ist deshalb genauer, jedoch langsamer als ein binär arbeitender Interpreter.

BASIC heißt: **B**eginners **a**ll-purpose **s**ymbolic **i**nstruction **c**ode

das bedeutet soviel wie: Allzweck-Programmiersprache für Anfänger

ERKLÄRUNG

BASIC ZEILEN-FORMAT

Eine **BASIC**-Zeile, die programmiert werden soll, enthält

1. eine Zeilennummer (ganzzahlig zwischen 0 und 32767) und
2. das **BASIC**-Befehlswort (in Grossbuchstaben) und
3. die Daten, auf die sich der Befehl bezieht und
4. die Taste [RETURN] als Abschlusszeichen

Die Punkte 2 und 3 werden zusammen häufig als "Statement" bezeichnet.

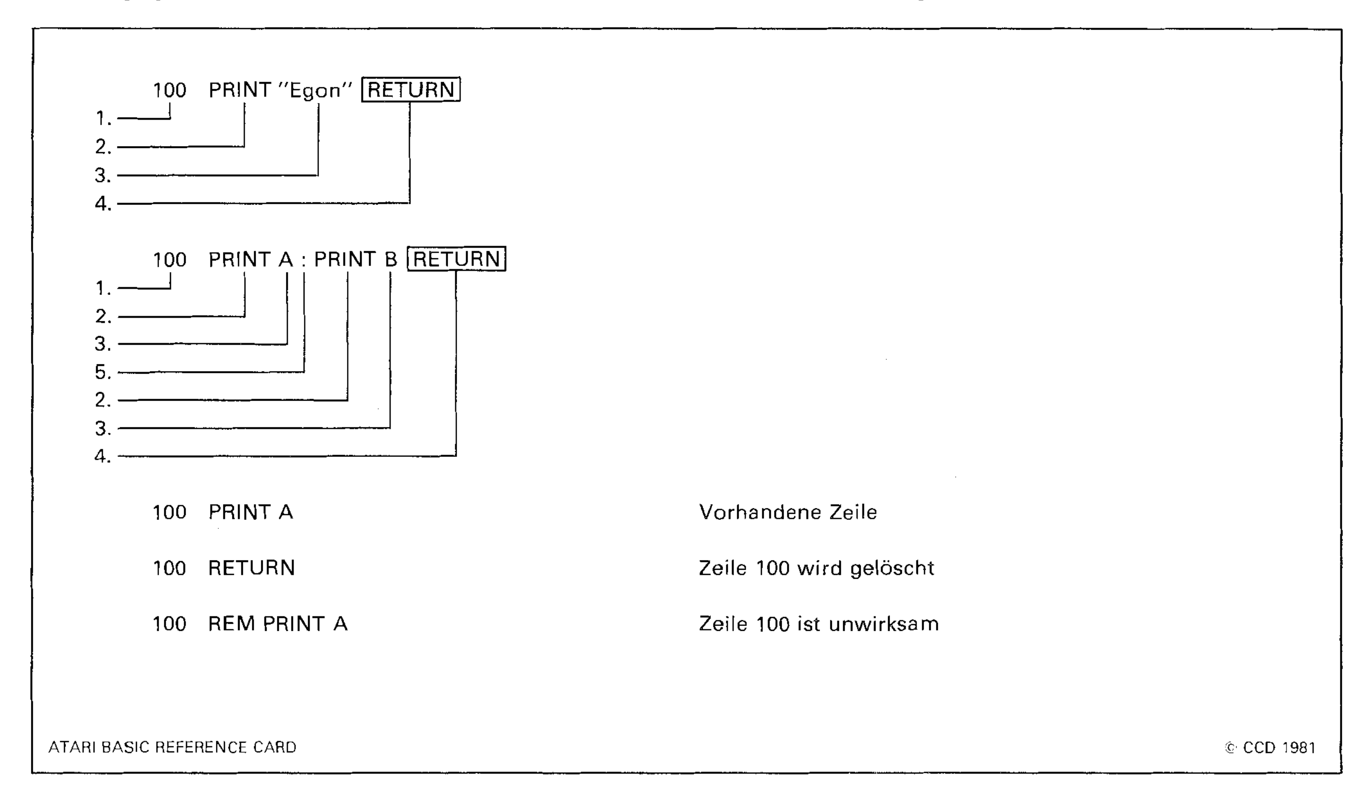
Eine **BASIC**-Zeile kann

5. mehrere Statements enthalten, die durch einen Doppelpunkt getrennt werden müssen und
6. bis zu drei Bildschirmzeilen lang sein

Soll die Zeile sofort ausgeführt werden, aber nicht programmiert werden, so entfällt bei der Eingabe die Zeilennummer. Um eine **BASIC**-Zeile zu löschen, wird nur die entsprechende Zeilennummer eingegeben. Alle **BASIC**-Statements die hinter einem **REM**-Statement stehen, werden nicht ausgeführt (siehe **REM**).

BEISPIELE

BASIC ZEILEN-FORMAT



BETRIEBSARTEN

1. Die Direkteingabe beginnt direkt mit dem **BASIC**-Befehlswort und wird sofort ausgeführt ohne, daß die Eingabe gespeichert wird. Sie wird mit der Taste [RETURN] abgeschlossen.

2. Die Programmeingabe beginnt für jede Eingabe mit der Zeilennummer, unter der die Eingabe von **BASIC** abgespeichert wird. Die **BASIC**-Zeile wird mit der Taste [RETURN] abgeschlossen.

3. Die Programmausführung beginnt mit der Eingabe von **RUN** oder **GOTO** mit Zeilennummer sowie der Taste [RETURN]. Das Programm wird nach steigender Zeilennummer abgearbeitet.

4. Die Bildschirmdarstellung schreibt von der Tastatur nur auf den Bildschirm, ohne daß die Eingabe direkt ausgeführt oder programmiert wird. Mit der Taste [RETURN] beginnt eine neue Zeile. Der Rechner meldet sich ohne Programmkassette in dieser Betriebsart ("**MEMO PAD**"), oder wenn von **BASIC** aus der Befehl **BYE** eingegeben wird.

BETRIEBSARTEN

1.	PRINT "A"	Sofortausdruck A
2.	100 PRINT "A"	Abspeicherung unter Zeile 100 LIST 100 PRINT "A"
3.	RUN	Ausführung eines abgespeicherten Programmes
4.	BYE	ATARI COMPUTER – MEMO PAD (zurück zu **BASIC** durch drücken der Taste [SYSTEM RESET])

ERKLÄRUNG

BOOTING

Automatisches Laden eines Maschinenprogramms beim Einschalten des Computers ("Urlader").

1. Der Computer ist ausgeschaltet.

2. Die Diskettenstation wird mit eingelegter Systemdiskette eingeschaltet und das Ausgehen der Busy-Lampe abgewartet.

3. Wird nun der Computer eingeschaltet, so stellt das Betriebssystem fest, daß die Diskettenstation betriebsbereit ist. Zum Booten eines Maschinenprogramms von der Kassette muß am Kassettenrekorder die Taste [PLAY] gedrückt sein. Während des Einschalten des Computers muß die Funktionstaste [START] gedrückt werden. Danach wird wie beim normalen Laden der Kassette fortgefahren (siehe Befehl **CLOAD**).

4. Obwohl eine Programmkassette eingesteckt sein kann, werden automatisch von der Diskette die Datei DOS.SYS und, falls vorhanden, ebenfalls automatisch die Datei AUTORUN.SYS geladen. Letztere kann Daten enthalten oder ein Maschinenprogramm, das nur geladen oder auch danach auch gestartet werden kann, wie dies bei vielen Programmen, die von der Diskette automatisch starten der Fall ist.

5. Ist AUTORUN.SYS nicht vorhanden, so wird an die Programmkassette übergeben, wenn eine solche eingesteckt ist.

6. Ohne Programmkassette und ohne die Datei AUTORUN.SYS wird die Datei DOS.DUP ebenfalls geladen und dann in das Menue gesprungen.

BOOTING

Mit dem folgenden Beispiel für AUTORUN.SYS wird erreicht, daß nach dem booten von DOS.SYS die Datei DOS.DUP ebenfalls automatisch geladen wird, ohnen daß in BASIC "DOS" eingegeben werden mußte.

```
100  FOR X=15000 TO 15010
200  READ W : POKE X , W
300  NEXT X
400  DATA 162 , 0 , 142 , 68 , 2 , 232 , 134 , 9 , 108 , 10 , 0

RUN
DOS
"SELECT ITEM OR RETURN FOR MENU"
K Taste [RETURN]
"SAVE-GIVE FILE, START, END(, INIT, RUN)"
AUTORUN.SYS , 3A98 , 3AA2 , , 3A98
```

Computer ausschalten und wieder einschalten.
Das System wird neu eingebootet und geht bei eingestecktem BASIC sofort zum DOS-Menue über.
(Beispiel vom ATARI Disk Operation System II Reference Manual, S. 68)

ERKLÄRUNG



Zur Unterbrechung der Programmausführung in diesem Augenblick.

Die Taste

1. unterbricht die Programmausführung im Augenblick des Drückens,
2. löscht das Programm und die Variablen nicht,
3. lässt die Datenkanäle, Tongeneratoren und den Grafikmodus unverändert,
4. kann nicht programmiert werden (siehe Befehl **STOP**),
5. kann wieder rückgängig gemacht werden, damit das Programm weiterläuft (siehe Befehl **CONT**),
6. gibt nach der Programmunterbrechung die Zeile an, in der das Programm angehalten wurde,
7. gibt nach durchgeführter Programmunterbrechung die Tastatur wieder für den Benutzer frei,
8. kann durch **POKE 16,112** abgeschaltet werden,
9. kann durch **POKE 16,192** oder Drücken der Taste SYSTEM RESET wieder eingeschaltet werden,
10. kann mit **PRINT PEEK (16)** abgefragt werden

BEISPIELE

TASTE [BREAK]

1. Taste [BREAK] — STOPPED AT LINE . . .

ERKLÄRUNG

BYE (B.)

Zum Verlassen der Programmiersprache **BASIC**.

Der Befehl

1. führt zum Verlassen von **BASIC** und bringt den Computer in die Betriebsart "Bildschirmdarstellung",
2. kann rückgängig gemacht werden durch drücken der Taste [SYSTEM RESET],
3. löscht das Programm und die Variablen nicht,
4. wird in der Regel direkt eingegeben,
5. kann auch programmiert werden.

BEISPIELE

BYE (B.)

	Eingabe	Ausgabe
1.	100 PRINT "TESTPROGRAMM" BYE	 ATARI COMPUTER – MEMO PAD (Schreiben auf dem Bildschirm. Programmieren nicht möglich)
2.	TASTE [SYSTEM RESET]	READY
3.	LIST	100 PRINT "TESTPROGRAMM"
4.	BYE	
5.	100 BYE	

ERKLÄRUNG



Zum Umschalten von Gross- auf Kleinbuchstaben und umgekehrt.

1. Beim Einschalten des Computers werden Grossbuchstaben dargestellt.

2. Wird bei eingeschalteten Grossbuchstaben die Taste [CAPS/LOWR] gedrückt, erfolgt die Umschaltung auf Kleinbuchstaben.

3. Wird bei eingeschalteten Kleinbuchstaben beim Drücken einer Buchstabentaste gleichzeitig die Taste [SHIFT] gedrückt, so wird dieser Buchstabe (wie bei einer Schreibmaschine) als Grossbuchstabe ausgegeben.

4. Werden bei eingeschalteten Kleinbuchstaben die Tasten [SHIFT] und [CAPS/LOWR] gleichzeitig gedrückt, erfolgt die Umschaltung auf Grossbuchstaben.

BEISPIELE

TASTE [CAPS/LOWR]

1.	ABCDEF	ABCDEF
2.	ABC [CAPS/LOWR] def	ABCdef
3.	[SHIFT] und O, dann tto	Otto
4.	uvw [SHIFT] und [CAPS/LOWR] XYZ	uvwXYZ

ERKLÄRUNG

CHR$

Zum Umwandeln einer ATASCII-Codezahl in eine String-Variable.

Die Funktion

1. findet das Zeichen, dessen ATASCII-Codezahl der Dezimalzahl (die zwischen 0 und 255 liegen muß) entspricht,
2. macht aus diesem Zeichen eine String-Variable,
3. darf in einer Vergleichsoperation nur einmal vorkommen.

(siehe Befehl **ASC** und ATASCII-Code)

BEISPIELE

CHR$

	Programm	Ergebnis
1.	100 DIM A$(2)	
	200 A$=CHR$(65)	
	300 PRINT A$	A
	400 PRINT CHR$(65)	A
	500 PRINT CHR$(65+256)	A ergibt keine Fehlermeldung, sondern es werden nur die 8 niederwertigsten Bits verwendet.
2.	PRINT ASC(CHR$(65))	65 Rückwandlung in den ATASCII-Code
3.	100 DIM A$(4) , B$(4)	
	200 A$="OTTO" : B$="EGON"	
	300 IF A$>B$ THEN PRINT"OK"	OK
	400 IF CHR$(79)>CHR$(69) THEN PRINT"AUCH OK"	Wird ohne Fehlermeldung übergangen

ERKLÄRUNG

Taste CLEAR

Zum Löschen des Bildschirmes.

Die Taste

1. löscht den Inhalt des Bildschirmes, wenn gleichzeitig
   a. die Taste [SHIFT] oder
   b. die Taste [CTRL]
   gedrückt wird,

2. kann programmiert werden, wenn der Befehl **PRINT** mit den
   a. Tasten [ESC] [SHIFT] oder den
   b. Tasten [ESC] [CTRL]
   verwendet wird,

3. kann im Programm in ihrer Wirkung durch
   a. Graphics 0 bzw. Gr. 0 oder
   b. PRINT CHR$(125)
   ersetzt werden, wenn der zu benutzende Drucker die grafischen Zeichen zu Punkt 2 nicht besitzt.

BEISPIELE

Taste CLEAR

Bildschirm wird gelöscht mit

1a.	Tasten [SHIFT] und [CLEAR]	SHIFT
1b.	Tasten [CTRL] und [CLEAR]	CTRL
2a.	100 PRINT"[ESC] [SHIFT] und [CLEAR]"	Befehl PRINT
2b.	100 PRINT"[ESC] [CTRL] und [CLEAR]"	Befehl PRINT
3a.	100 GRAPHICS 0 100 GR. 0	Grafikmodus 0
3b.	100 PRINT CHR$(125)	Befehl PRINT mit Umwandlung des ATASCII-Codes 125 für das Bildschirmlöschen

ERKLÄRUNG

CLOAD (CLOA.)

Zum Laden eines Programmes von der Kassette in den Computer-Speicher (RAM).

Der Befehl

1. löscht ein im RAM-Speicher stehendes Programm samt Variablen,
2. lädt ein Programm in der übersetzten Form (siehe Befehle **LOAD** und **ENTER**),
3. lädt nur Programme, die mit **CSAVE** gespeichert worden sind,
4. kann nicht durch den Befehl **RUN"C :** " ersetzt werden,
5. kann nicht mit einem Programmnamen versehen werden,
6. wird in der Regel direkt eingegeben,
7. kann auch programmiert werden
8. benötigt kein OPEN oder **CLOSE**

CLOAD (CLOA.)

Bedienung des Kassettenrekorders:

1. Kassette einlegen und mit Hilfe des Zählwerkes das Band an die gewünschte Stelle bringen,
2. Eingabe **CLOAD** und Taste [RETURN] ,
3. Lautsprecher in der Tastatur ertönt einmal,
4. am Kassettenrekorder Taste [PLAY] drücken,
5. Eingabe Taste [RETURN] ,
6. das Band wird gestartet und das Programm geladen.
7. Der Ladevorgang ist beendet, wenn **READY** erscheint.

zu

5. CLOAD

6. 100 CLOAD

ERKLÄRUNG

CLOG

Zur Berechnung des dekadischen Logarithmus einer Zahl.

Die Funktion

1. errechnet den Logarithmus zur Basis 10

    a. der Zahl,
    b. der Variablen oder
    c. des zu berechnenden Wertes

    die/der in der Klammer hinter **CLOG** steht,

2. erwartet einen positiven Klammerinhalt.

1a.	100	PRINT CLOG(100)	2
1b.	100	C = 100 : PRINT(C)	2
1c.	100	PRINT CLOG(200 − 100)	2
2a.	100	PRINT CLOG(1)	2E−10, Rundungsfehler abfangen, eigentlich 0
2b.	100	PRINT CLOG(0)	abfangen, da nicht definiert
2c.	100	PRINT CLOG(ABS(−100))	2

ERKLÄRUNG

CLR

Zum Löschen der Variablen.

Der Befehl

1. löscht alle vorher dimensionierten Variablen,
2. setzt alle nicht dimensionierten Variablen auf 0, löscht ihre Namen jedoch nicht,
3. lässt die Programmzeilen unbeeinflußt,
4. kann programmiert werden,
5. kann direkt eingegeben werden.

BEISPIELE

CLR

1.	100 DIM A$(4) : A$ = "Otto"	
	100 PRINT A$	Otto
	CLR : PRINT A$	ERROR – 9 (A$ ist gelöscht)
2.	100 A = 10 : PRINT A	10
	CLR : PRINT A	0
3.	100 A = 10 : PRINT A	10
	CLR LIST	
		100 A = 10 : PRINT A
4.	100 CLR	
5.	CLR	

ERKLÄRUNG

COLOR (CO.)

Zum Zuordnen der Farbe zu einem bestimmten Farbregister für die auf **COLOR** folgenden Befehle **PLOT** und **DRAWTO**.

Der Befehl

1. bereitet die Ausgabe an das Grafikfenster für die Befehle **PLOT** und **DRAWTO** vor; die eigentliche Ausgabe erfolgt an der durch sie bezeichneten Stelle, sobald **PLOT** oder **DRAWTO** durchgeführt werden,
2. erwartet hinter **COLOR** eine Zahl zwischen 0 bis 255,
3. Grafikmodus 0:
   gibt alle Zeichen des ATASCII-Codes deren Wert hinter **COLOR** angegeben wurde mit der Farbe von **SE. 2** und der Helligkeit von **SE. 1** aus,
4. Grafikmodi 1 und 2:
   es werden alle eingegebenen Zeichen zu Großbuchstaben und Ziffern:
   a. Großbuchstabe/Ziffer : Farbe aus SE. 0,
   b. Kleinbuchstabe : Farbe aus SE. 1,
   c. Inv. Großb./Ziffer : Farbe aus SE. 2,
   d. Inv. Kleinb. : Farbe aus SE. 3,
5. Grafikmodi 3 bis 8 und 10:
   Grafikpunkte mit der Farbe aus dem Farbregister ("Farbeimer"), das dem COLOR-Wert ("Pinsel") durch den Grafikmodus zugeordnet worden ist,
6. Grafikmodus 9:
   Angabe der Helligkeitsstufe für die Farbe aus **SE. 4**,
7. Grafikmodus 11:
   Angabe der Farbkennzahl mit der Helligkeit aus **SE. 4**

COLOR (CO.)

Modus	Ausgabe von	Beispiel
0	blaues, großes A	100 GR. 0 : CO. 65 : PL. 5 , 5
1	grünes, großes A	100 GR. 1 : CO. 65+32 : PL. 5 , 5 : REM KLEINES A*
2	rotes, großes A	100 GR. 1 : CO. 65+32+128 : PL. 5 , 5 : REM KL. INV. A*
3 – 7	orangef. Grafikp.	100 GR. 3 : CO. 1 : PL. 5 , 5 : REM ANFANGSF. ORANGE
8	dunkelbl. Grafikp.	100 GR. 9 : SE. 2 , 0 , 0 : CO. 1 : PL. 5 , 5 : REM ANFANGSF.
9	hellbl. Grafikp.	100 GR. 9 : SE. 4 , 9 , 0 : CO. 8 : PL. 5 , 5 dunkelste Farbe nach SE. 4 200 GOTO 200
10	orangef. Grafikp.	100 GR. 10 : CO. 4 : PL. 5 , 5 Anfangsfarbe aus SE. 0 200 GOTO 200
11	hellroter Grafikp.	100 GR. 11 : SE. 4 , 0 , 10 : CO. 3 : PL. 5 , 5 Helligkeit aus SE. 4 200 GOTO 200

* siehe ATASCII-Code

ERKLÄRUNG

CONT (CON.)

Zur Fortsetzung eines unterbrochenen Programmes.

Der Befehl setzt die Ausführung eines unterbrochenen Programmes fort, wenn

1. das Programm durch die Taste [BREAK] unterbrochen wurde oder
2. bei der Programmausführung wegen des Erreichens eine STOP- oder END-Befehls das Programm unterbrochen wurde und das Ende des Programmes noch nicht erreicht war.
3. Der Befehl kann auch programmiert werden.

Die Programmaufführung wird in der folgenden **BASIC**-Zeile wieder aufgenommen. Wurde das Programm in der Mitte einer Zeile unterbrochen, so werden bei der Weiterführung die restlichen Befehle dieser Zeile nicht mehr ausgeführt.

BEISPIELE

CONT (CON.)

1. Taste [BREAK]
   STOPPED AT LINE . . .
   CONT

2.
```
100  PRINT "A" : STOP : PRINT "B"
110  PRINT "C"
RUN                                    A
                                       STOPPED AT LINE 100
CONT                                   C
```

3.
```
100  POSITION 2 , 23 : PRINT "CONT"
110  POSITION 2 , 21 : POKE 764 , 12
120  STOP
130  PRINT "HIER IST ZEILE 130"
RUN                                    STOPPED AT LINE 120
                                       CONT
                                       HIER IST ZEILE 130
                                       READY
```

ERKLÄRUNG

COS

Zum Errechnen des Kosinus.

Die Funktion

1. errechnet den Kosinus
   a. der Zahl,
   b. der Variablen oder
   c. des zu berechnenden Wertes

die/der in der Klammer hinter **COS** steht,

2. ergibt nur Werte zwischen +1 und −1,

3. kann den Winkel im Bogenmaß oder in Grad verarbeiten (siehe Befehle **DEG** und **RAD**).

1a.	100 DEG	Umschaltung auf Grad
	200 PRINT COS(90)	0
	300 PRINT COS(720)	1
1b.	100 A=90 : PRINT COS(A)	0
1c.	100 PRINT COS(90−180)	0
3.	100 RAD	Umschaltung auf Bogenmaß
	200 PI=3.1415926535	
	PRINT COS(PI)	−1

ERKLÄRUNG

CSAVE (CS.)

Zum Speichern eines Programmes vom Computer-Speicher (RAM) auf die Kassette.

Der Befehl

1. verändert ein im RAM-Speicher stehendes Programm nicht,
2. speichert das Programm in übersetzter Form auf das Band (siehe Befehle **SAVE** und **LIST**),
3. kann nicht mit einem Programmnamen versehen werden,
4. wird in der Regel direkt eingegeben,
5. kann auch programmiert werden,
6. speichert schneller als **SAVE"C:"**, weil die Pausen zischen den Blöcken kürzer sind,
7. benötigt kein **OPEN** oder **CLOSE**.

BEISPIELE

CSAVE (CS.)

Bedienung des Kassettenrekorders:

1. Kassette einlegen und mit Hilfe des Zählwerkes das Band an die richtige Stelle bringen,
2. Eingabe **CSAVE** und Taste [RETURN]
3. der Lautsprecher der Tastatur ertönt zweimal,
4. Kassettenrekorder-Tasten [REC] und [PLAY] gleichzeitig drücken,
5. Eingabe Taste [RETURN]
6. das Programm wird gestartet und das Programm wird gespeichert.
7. Der Schreibvorgang ist beendet, wenn **READY** erscheint.

zu

4. CSAVE
5. 100 CSAVE

ERKLÄRUNG



Zum Umschalten der Funktion einiger Tasten (zweite Umschaltebene).

Wird die Taste CTRL gleichzeitig gedrückt mit

1. der Taste 1, so wird die Bildschirmausgabe und die Programmausführung unterbrochen,
2. der Taste 1, wenn vorher das Programm damit angehalten worden war, so wird die Bildschirmausgabe und die Programmausführung fortgesetzt,
3. der Taste 2, so ertönt der Summer in der Tastatur,
4. der taste 3, ergibt eine Fehlermeldung **EOF** (**END OF FILE**) Meldung,
5. mit einer Buchstabentaste oder mit einer Taste für Punkt, Komma oder Semikolon, so wird das entsprechende Pseudografik-Zeichen dargestellt,
6. mit der Taste CAPS/LOWR, so wird die Tastatur auf Dauer auf Pseudografik-Zeichen umgeschaltet,
7. der Taste CLEAR, so wird der Bildschirm gelöscht,
8. mit einer der Cursorsteuertasten, so wird der Cursor entsprechend bewegt,
9. mit der Taste INSERT, so wird an der Cursorposition ein Leerzeichen eingeschoben; der Rest der logischen Zeile rückt nach rechts,
10. mit der Taste DELETE BACK'S, so wird das Zeichen an der Cursorposition entfernt; der Rest der logischen Zeile rückt nach links
11. mit der Taste TAB, so wird die Tabulatormarke an der Cursorposition gelöscht.

BEISPIELE



1.	Tasten [CTRL] und 1	Programm hält an
2.	Tasten [CTRL] und 1 nochmals	Programm läuft weiter
3.	Tasten [CTRL] und 2	Summer ertönt
4.	Tasten [CTRL] und 3	ERROR - 136 (Bei Benützung von DOS mit dem Befehl C → E : , D : TEST kopiert aus dem Editor Textspeicher in die Datei D : TEST. Eingabe mit [CTRL] 3 abschließen.
5.	Tasten [CTRL] und R	——
6.	Tasten [CTRL] und [CAPS/LOWR]	Umschaltung auf Pseudografik
7.	Tasten [CTRL] und [CLEAR]	Bildschirm wird gelöscht
8.	Tasten [CTRL] und [→]	Cursor eine Position nach rechts
9.	Tasten [CTRL] und [INSERT]	Leerzeichen einschieben
10.	Tasten [CTRL] und [DELETE BACK'S]	Zeichen entfernen
11.	Tasten [CTRL] und [TAB]	Tabulatormarke wird gelöscht

TASTEN CURSOR

Zur Kennzeichnung des Ausgabeortes des nächsten Zeichens auf dem Bildschirm.

Der Cursor kann über 4 Steuertasten durch die Zeilen und Spalten des Bildschirmes geführt werden, ohne daß der Bildschirminhalt verändert wird.

1. [CTRL] [→] führt den Cursor eine Spalte nach rechts. War der Cursor in der letzten Spalte, springt er in die erste Spalte der folgenden Zeile.

2. [CTRL] [←] führt den Cursor eine Spalte nach links. War der Cursor in der ersten Spalte, springt er in die letzte Spalte derselben Zeile.

3. [CTRL] [ ] führt den Cursor eine Zeile nach oben. War der Cursor in der obersten Zeile, bleibt er in derselben Spalte und springt in die unterste Zeile.

4. [CTRL] [ ] führt den Cursor eine Zeile nach unten. War der Cursor in der untersten Zeile, bleibt er in derselben Spalte und springt in die oberste Zeile.

TASTEN CURSOR

a. Cursorbewegungen können programmiert werden:

```
100 PRINT " [ESC] [CTRL] [→] "
```

b. Der Cursor kann unsichtbar gemacht werden:

```
100 POKE 752,1
```

c. Der Cursor kann wieder sichtbar gemacht werden:

```
100 POKE 752,0
```

ERKLÄRUNG

Datei

Eine Datei (engl. "File") ist eine Ansammlung von Daten, auf die gemeinsam über die Dateibeschreibung zugegriffen werden kann.

1. Es werden zwei Arten von Dateien unterschieden:
   a. Programm-Dateien, die Befehle für den Computer enthalten,
   b. Daten-Dateien, die Daten enthalten, die ein Programm verwendet.

2. Der Dateiname ("Filename") dient zur Bezeichnung einer Datei und darf höchstens 8 Zeichen enthalten; das erste Zeichen muß ein Buchstabe sein (zulässige Zeichen sind A – Z und 0 – 9).

3. Die Ergänzung des Dateinamens ("Filename-Extender") besteht aus einem Punkt und zwischen 0 und 3 Zeichen. Durch sie ist es möglich, Dateien mit identischen Dateinamen zu unterscheiden. Sie kann jedoch auch weggelassen werden.

4. Die Dateibeschreibung ("File-Specification") dient zur Identifizierung einer Datei und besteht aus
   a. dem Kennzeichen des Gerätes, das benutzt werden soll,
   b. dem Dateinamen und
   c. der Ergänzung des Dateinamens,
   die gemeinsam in Anführungszeichen gesetzt werden müssen.

5. Die Dateibeschreibung kann in Form einer String-Variablen haben.

   (siehe Befehle **ENTER, LIST, SAVE, LOAD** und **RUN**).

BEISPIELE

Datei

1a.	RUN"D : PROGRAMM"	Programm-Datei
1b.	100 OPEN#1 , 8 , 0 ,"D : DATEN"	einfaches Dateiprogramm, das Zahlen, die kleiner als 256
	200 PRINT"WELCHE ZAHL " ; : INPUT Z	sind, in der Datei "DATEN" ablegt.
	300 IF Z 255 THEN END	
	400 PUT#1 , Z : GOTO 200	
2.	LOAD"D : OTTO17"	diese beiden Namen werden unterschieden
3.	LOAD"D : OTTO17.NEU"	
4a.	D1 :	Kennzeichen der Diskette 1
4b.	OTTO17	Dateiname
4c.	.NEU	Version "NEU" von "OTTO17"
5a.	DIM a$(15) : A$="D1 ; OTTO17.NEU"	Dateibeschreibung als String
	RUN A$	geht auch mit RUN
5b.	SAVE A$	geht auch mit SAVE
5c.	LOAD A$	geht auch mit LOAD
5d.	LIST A$	geht auch mit LIST
5e.	ENTER A$	geht auch mit ENTER

ERKLÄRUNG

DEG

Zum Umschalten der Winkelfunktion auf Grad.

Der Befehl

1. schaltet die Winkelfunktionen auf Grad (Kreisumfang entspricht 360°),

2. wird aufgehoben, wenn die Taste [SYSTEM RESET] gedrückt wird (Umschaltung auf Bogenmaß).

BEISPIELE

1.	DEG	Umschaltung auf Grad
	PRINT SIN(90)	1 (Eingabe als Grad verarbeitet)
	Taste [SYSTEM RESET]	
	PRINT SIN(90)	0.8939970243 (Eingabe als Bogenmaß verarbeitet.

Taste DELETE BACK'S

Zum Entfernen von Zeichen und Zeilen.

Wird die Taste [DELETE BACK'S] gleichzeitig gedrückt mit

1. der Taste [CTRL], so wird das Zeichen unter dem Cursor gelöscht; der Rest der logischen Zeile rückt eine Position nach links,

2. der Taste [SHIFT], so wird die Zeile in der der Cursor steht gelöscht; der Rest des Bildschirminhaltes rückt eine Zeile nach oben.

Wird die Taste [DELETE BACK'S] allein gedrückt,

3. springt der Cursor eine Position zurück und löscht das Zeichen vor der alten Cursorposition; der Rest der logischen Zeile wird nicht herangerückt.
   Steht dabei der Cursor in der ersten Spalte, so wird
   a. der Cursor nicht weiter nach links geschoben, wenn der Anfang der logischen Zeile erreicht ist,
   b. der Cursor in die letzte Spalte der nächsthöheren Zeile gerückt und das dortige Zeichen gelöscht; dies geht so lange, bis der Beginn der logischen Zeile erreicht ist.

BEISPIELE

Taste DELETE BACK'S

1. AB[C]DEF — Cursor auf C
   zweimal Tasten [CTRL] und [DELETE BACK'S]
   AB[E]F — zwei Zeichen gelöscht

2. ABCDEF
   G[H] IJKL — Cursor auf H
   MNOPQR
   STUVWX
   ______________________ — Bildschirmunterkante

   zweimal Tasten [SHIFT] und [DELETE BACK'S]
   ABCDEF
   S[T]UVWX — zwei Zeilen gelöscht

   ______________________ — Bildschirmunterkante

ERKLÄRUNG

DIM (DI.)

Zum Bereitstellen des benötigten Speicherraumes für String-Variablen, sowie für einfach und doppelt indizierte Variablen.

Der Befehl

1. * reserviert für die String-Variable, deren Name hinter **DIM** steht, die Anzahl Zeichenpositionen, die in der Klammer hinter dem Variablennamen angegeben ist,
2. ** reserviert für die einfach indizierte Variable, deren Name hinter **DIM** steht, eine Eintragung mehr in eine eindimensionale Liste, als in der Klammer hinter dem Variablennamen angegeben ist,
3. ** reserviert für die doppelt indizierte Variable, deren Name hinter **DIM** steht, je eine Eintragung mehr in eine zweidimensionale Liste, als in der Klammer hinter dem Variablennamen angegeben ist; dabei gibt die erste Zahl in der Klammer die Anzahl der Zeilen und die zweite die der Spalten an,
4. sollte immer am Programmbeginn stehen,
5. füllt die dimensionierten Variablen nicht automatisch mit 0,
6. kann mehrere Variablen (durch Komma getrennt) gleichzeitig dimensionieren,
7. führt bei wiederholtem Gebrauch zur Fehlermeldung 9 (siehe Befehl **CLR**).

* Festlegung der Zeichenpositionen von 1 bis zum angegebenen Klammerwert

** Festlegung der Zahlenpositionen von 0 bis zum angegebenen Klammerwert

BEISPIELE

1.	100 DIM A$(10)	Stringvariable mit 10 Zeichen
2.	100 DIM A(3)	einfach indizierte Variable für eindimensionale liste mit 4 Listenplätzen [A(0) ist auch verwendbar]
3.	100 DIM B(2 , 5)	doppelt indizierte Variable für zweidimensionale Liste mit 3 Zeilen und 6 Spalten
5.	100 DIM C(4)	eindimensionale Liste mit 5 Plätzen
	200 FOR X=0 TO 4	Schleife von 0 bis 4 und
	300 C(X)=0	darin den X-ten Platz auf Null setzen
	400 NEXT X	
6.	100 DIM C$(10) , E(2 , 3)	Dimensionieren einer Stringvariablen mit 10 Stellen und einer zweidimensionalen Liste mit 3 Zeilen und 4 Spalten
7.	100 DIM Z$(10)	Z$ wird zweimal ohne CLR dimensioniert
	200 DIM Z$(3)	ERROR 9 AT LINE 200

DOS (DO.)

Zum Übergang von **BASIC** zum **DOS** (Diskettenbetriebssystem).

Der Befehl

1. setzt voraus, daß **DOS** "eingebootet" wurde (siehe booting),
2. führt von **BASIC** zu **DOS** und schreibt das DOS-Programm-Menü auf den Bildschirm,
3. führt von **BASIC** in die Betriebsart Bildschirmdarstellung("Memo Pad"), wenn **DOS** nicht eingebootet worden ist,
4. wird in der Regel direkt eingegeben,
5. kann auch programmiert werden,
6. kann rückgängig gemacht werden durch drücken der Taste [SYSTEM RESET] ("READY"),
7. kann rückgängig gemacht werden, wenn bei dargestelltem DOS-Menü die Tasten B und [RETURN] gedrückt werden.

(siehe Handbücher für Diskettenstation und -betriebssystem)

BEISPIELE

DOS (DO.)

zu

4. DOS

5. 100 DOS

DRAWTO (DR.)

Zum Zeichnen einer Linie

Der Befehl

1. wirkt in allen Grafikmodi,
2. zeichnet eine Linie zu der hinter **DRAWTO** angegebenen Position im Grafikfenster,
3. enthält die horizontale Position an der ersten Stelle hinter **DRAWTO** [X] , Y
4. enthält die vertikale Position an der zweiten Stelle hinter **DRAWTO** X , [Y]
5. zeichnet die Linie in der Farbe und Helligkeit, die durch den davorliegenden **COLOR**-Befehl bestimmt worden ist,
6. ergibt in den Grafikmodi 3-11 nur dann eine sichtbare Linie, wenn **COLOR** ungleich 0 ist (COLOR 0 = identisch mit Hintergrundfarbe),
7. zeichnet von der augenblicklichen Grafik-Cursorposition aus,
8. verwendet als Nullpunkt für die Positionsangaben die linke, obere Bildschirmecke (X=0 , Y=0).

(siehe Befehle **PLOT** und **COLOR**)

BEISPIELE

DRAWTO (DR.)

100 GRAPHICS 3	Grafikmodus 3
	es werden die Anfangsfarben verwendet
200 COLOR 1	Farbe ist orange
300 PLOT 0 , 5 : DRAWTO 10 , 5	orangefarbene Linie in Zeile 5
400 COLOR 2	Farbe 2 ist hellgrün
500 DRAWTO 10 , 0 : DRAWTO 10 , 10	hellgrüne Linie
600 COLOR 3	Farbe 3 ist dunkelblau
700 DRAWTO 0 , 10 : DRAWTO 0 , 0	dunkelblaue Linie
800 GOTO 800	Schleife zum verhindern der automatischen Rückschaltung auf Grafikmodus 0 durch das Programmende

100 GRAPHICS 0	Textmodus 0
200 POKE752 , 1	Cursor aus
300 COLOR 65 : REM ATASCII VON A	Buchstabe großes A als Linie
400 PLOT 2 , 0 : DRAWTO 5 , 3	
500 PRINT"OK"	

```
A
 A
  A
   OK
READY
```

ERKLÄRUNG

END

Zur Beendigung des Programmes an dieser Stelle.

Der Befehl

1. beendet die Programmausführung an dieser Stelle,
2. schliesst alle Datenkanäle,
3. schaltet alle Tongeneratoren aus,
4. schaltet den Bildschirm in den Grafikmodus 0,
5. sollte immer zwischen Haupt- und Unterprogramm stehen,
6. wird am Ende des Programmes nicht benötigt,
7. wird in der Regel programmiert,
8. kann aber auch als Direkteingabe zum Schliessen der Datenkanäle und zum Ausschalten der Tongeneratoren verwendet werden,
9. gibt nach der Beendigung des Programmes die Zeilennummer nicht an (siehe Befehl **STOP**),
10. gibt nach der Programmbeendigung die Tastatur wieder für den Benutzer frei ("READY").

BEISPIELE

END

zu

1. 100 PRINT A : END

5.
```
100 PRINT " Hauptpr. "
200 GOSUB 400
300 END
400 PRINT " Unterpr. "
500 RETURN
```

7. 900 END

8. END

10. READY

ENTER (E.)

Zum Lesen eines Programmes in den Speicher.

Der Befehl

1. löscht ein im Speicher stehendes Programm samt Variablen nicht (ermöglicht das Zusammenfügen mehrerer Programme),
2. liest das Programm im Klartext und übersetzt es beim Lesen (siehe Befehle **LOAD** und **CLOAD**),
3. liest nur Programme, die mit **LIST** gespeichert wurden,
4. kann a) vom Kassettenrekorder oder
   b) von der Diskette lesen,
5. kann mit einem Programmnamen versehen werden,
6. wird in der Regel direkt eingegeben,
7. kann auch programmiert werden.

Beim Zusammenfügen von Programmen sollten sich die verwendeten Zeilennummern wegen der besseren Übersicht nicht überlappen. Wird ein vorher mit **LIST** gespeichertes Programm mit dem Befehl **ENTER** gelesen, verhält sich der Computer so als würde Zeile für Zeile über die Tastatur eingegeben werden.

ENTER (E.)

zu

```
 1a.  100  PRINT"Zeile 100"
      300  PRINT"ZEILE 300"
      400  PRINT"ZEILE 400"
      LIST"C : " : NEW                    Programm auf Kassette speichern und im Computer löschen
 1b.  200  PRINT"ZEILE 200"
      400  PRINT"ENDE"                    neues Programm (1b) eingeben,
      ENTER"C : "                         laden des ersten Programms (1a)
      LIST
      100  PRINT"ZEILE 100"
      200  PRINT"ZEILE 200"               Zeile 200 wurde eingeschoben
      300  PRINT"ZEILE 300"
      400  PRINT"ENDE"                    Zeile 400 wurde ersetzt
 4a.  ENTER"C : "
 4b.  ENTER"D : NAME.TXT"                 hier ist der Name Vorschrift
 5a.  ENTER"C : PROGR1.BAS"
 5b.  ENTER"D : PROGR2.BAS"
 5c.  100  DIM N$(15) : N$="D : NAME.BAS"
      200  ENTER N$                       Dateibeschreibung als Stringvariable
 6.   ENTER"C : "
 7.   100  ENTER"C : "
```

ERROR

Fehlermeldung

Eine Fehlermeldung wird ausgegeben, wenn

1. bei der Programmeingabe in der BASIC-Zeile ein Syntax-Fehler vorliegt. Die fehlerhafte Zeile wird mit "**ERROR**" zwischen Zeilennummer und dem ersten Statement ausgegeben und an der Stelle, bis zu der kein Fehler vorlag, markiert. "**ERROR**" muß entfernt werden bevor die Zeile wieder eingegeben werden darf. Wird eine Programmzeile mit "**ERROR**" vom Programm verarbeitet, so erscheint die Fehlermeldung "**ERROR 17 AT LINE ...** "

2. bei der Programmdurchführung ein logischer Fehler auftritt, nach dem das Programm abgebrochen wird, wenn nicht mit dem Befehl **TRAP** eine Fehlerbehebung durchgeführt wird. Dabei wird die Fehlerkennzahl ausgegeben und angegeben, in welcher Zeile der Fehler auftrat.

3. bei einer Eingabe oder Ausgabe über ein Peripheriegerät ein bestimmtes Problem auftritt.

   (siehe Befehl **TRAP**)

2	Speicher voll	20	Kanalnummer falsch	143	Datenverkehr gestört (Prüfsumme)
3	Zahlenbereich falsch	21	falscher Ladebefehl	144	Disk. kann nicht gel./beschr. werden
4	mehr als 128 Variablen	128	mit BREAK abgebrochen	145	Fehler beim Prüflesen
5	String zu lang	129	Datei schon eröffnet	146	Funktion nicht vorgesehen
6	zu wenige Data	130	Gerät nicht bekannt	147	Speicher reicht nicht
7	Zahlenbereich falsch	131	nur Ausgaben möglich	160	falsche Diskettenst.-Nummer
8	Input in falsche Variable	132	unzulässige Eingabe	161	zu viele Dateien offen
9	DIM-Fehler	133	OPEN fehlt	162	Diskette voll
10	nicht ausführbar	134	Kanalnummer unzulässig	163	endgültiger Datenverlust
11	Zahlenbereich verlassen	135	nur Eingaben möglich	164	behebbarer Datenverlust
12	Zeile fehlt	136	Dateiende mit EOF	165	Dateiname fehlerhaft
13	kein passendes FOR	137	Datensatz verstümmelt	166	POINT-Angaben falsch
14	Zeile zu lang	138	Ausgabegerät antwortet nicht	167	Datei gesichert
15	GOSUB/FOR entfernt	139	Datenverkehr gestört (Systemf.)	168	Befehl unzulässig
16	passendes GOSUB fehlt	140	Lesefehler	169	Inhaltsverzeichnis voll
17	Speicherfehler	141	Cursorposition unzulässig	170	Datei nicht gefunden
18	Stringanfang falsch	142	Datenverkehr gestört (Formatf.)	171	POINT-Angaben ungültig
19	Programm zu lang *				

* Die Fehlermeldungen ab 19 treten im Dialog mit den Ausgabegeräten auf und werden in den entsprechenden Handbüchern dieser Geräte noch ausführlicher besprochen.

ERKLÄRUNG



Zur Programmierung der Bildschirmkommandos über den **PRINT**-Befehl.

Dazu gehören:

1. Cursorbewegungen
2. Bildschirmlöschen
3. Zeichen- und Zeileneinfügungen
4. Zeichen- und Zeilenentfernen
5. Tabulator-Positionen
6. Summer, EOF, Bildschirm "einfrieren"

Die Umschaltung gilt immer nur für ein Kommando. Die Taste muss deshalb jedesmal neu gedrückt werden.

BEISPIELE

TASTE [ESC]

1.	100 PRINT " [ESC] [CTRL] [→] "	Cursor 1 Position nach rechts
2.	100 PRINT " [ESC] [CTRL] u. [CLEAR] "	Bildschirm wird gelöscht
3.	100 PRINT " [ESC] [CTRL] u. [INSERT] "	1 Zeichen wird eingeschoben
4.	100 PRINT " [ESC] [TAB] "	Cursor zur nächsten Tab-Position
6a.	PRINT " [ESC] [CTRL] u. 2	Summer
6b.	LIST [CTRL] und 1	hält die Ausgabe des Programmlistings an. (nochmaliges [CTRL] u. 1 startet es wieder)

ERKLÄRUNG

EXP

Zur Berechnung von Potenzen der Zahl e.

Die Funktion

1. errechnet die Zahl e (2.71828179) hoch
   a. die Zahl,
   b. die Variable oder
   c. den zu berechnenden Wert

   die/der in der Klammer hinter **EXP** steht,

2. ist in einigen Fällen nur auf 6 Stellen genau (eventuell abrunden).

BEISPIELE

EXP

	Programm	Ergebnis
1a.	100 PRINT EXP(2)	7.38905599
1b.	100 A = 1 : PRINT EXP(A)	2.71828179
1c.	100 PRINT EXP(10−8)	7.38905599
2a.	100 PRINT EXP(0)	0.9999999998 (Rundungsfehler, richtig 1)
2b.	100 C = EXP(0) 200 C = INT(C*1E5+.5)/1E5 300 PRINT C	1 (Rundungsroutine für 5 Stellen hinter dem Komma)

ERKLÄRUNG

FOR (F.)/NEXT (N.)

Zum wiederholten Abarbeiten eines bestimmten Programmteiles innerhalb der sogenannten FOR/NEXT-Schleife ("Loop").

Der Befehl

1. gibt den Beginn der Programmschleife an ("FOR"),

2. setzt den Schleifenzähler auf den Anfangswert ("X=2"),

3. gibt den Endwert des Schleifenzählers an ("TO 8"),

4. gibt die Zählrichtung und die Größe der Zählschritte an ("STEP +2"); dieser Punkt darf weggelassen werden; in diesem Fall verwendet der Computer "STEP +1". Anfangswert, Endwert oder Zählschritt können Zahlen numerische Variablen oder arithmetische Ausdrücke sein. Der Zählschritt kann positiv oder negativ sein. Ist er gleich Null (0) führt dies zu einer Endlosschleife. Jede Schleife wird mindestens einmal durchlaufen (siehe Befehl **POP**).

5. gibt das Ende der Programmschleife an; nachdem der Schleifenzähler um einen Zählschritt weitergezählt hat, wird von hier aus zum ersten Befehl innerhalb der Schleife zurückgesprungen, wenn der Schleifenzähler den Endwert noch nicht überschritten hat ("NEXT X"),

6. beendet die Schleife, wenn der Schrittzähler den Endwert überschritten hat ("bedingter Sprung"),

7. verarbeitet auch ineinander verschachtelte Schleifen, wobei die innerste Schleife zuerst beendet wird.

FOR (F.)/NEXT (N.)

1.
```
100  FOR X=2 TO 8 STEP +2
200  PRINT X,
300  NEXT X
400  PRINT "ENDE"
500  END
```

Der Variablenname des Schleifenzählers muß wiederholt werden.

```
RUN
```

```
2          4          6          8
ENDE
READY
```

7.
```
100  FOR X=1 TO 3                        äußere Schleife
200  FOR Y=1 TO 2                        innere Schleife
300  PRINT "X= ";X;"    Y= ";Y           zu wiederholendes Programm
400  NEXT Y                              Schleifenende von Y zuerst
500  NEXT X                              und dann Schleifenende von X
RUN
```

```
X=1          Y=1
X=1          Y=2
X=2          Y=1
X=2          Y=2
X=3          Y=1
X=3          Y=2
```

ERKLÄRUNG

FRE

Zum Feststellen des freien Speicherraumes.

Die Funktion

1. errechnet die Anzahl der für Benutzerzwecke verfügbaren Speicherzellen,
2. benötigt in der Klammer hinter **FRE** immer eine "Dummy"-Variable,
3. wird in der Regel direkt eingegeben,
4. kann auch programmiert werden.

BEISPIELE

FRE

1.
```
PRINT FRE(0)
```
37902 (willkürlich gewähltes Beispiel)

2.
```
100 IF<FRE(0)=1000
THEN PRINT "ACHTUNG"
```

ERKLÄRUNG

GET (GE.)

Zur Eingabe eines Bytes über einen eröffneten Datenkanal.

Der Befehl

1. setzt einen mit **OPEN** eröffneten Datenkanal voraus (Kennzahl 4 oder 12),
2. muß mit # und der Nummer des Datenkanals versehen werden, über den gelesen werden soll,
3. liest ein Byte ein (8-Bit-Parallelwort, entspricht einer Zahl zwischen 0 und 255), das in der Variablen abgelegt wird, die hinter dem Komma angegeben ist.

(siehe Befehl **PUT**)

BEISPIELE

GET (GE.)

1. 100 OPEN #1 , 4 , 0 , "D : TEST"

2. 200 GET #1 , A : PRINT A

3. 100 GET #1 , ZIEL — legt das gelesene Byte in der Variablen Ziel ab

ERKLÄRUNG

GOSUB (GOS.)/RETURN (RET.)

Zum Aufruf eines in **BASIC** geschriebenen Unterprogrammes durch ein **BASIC**-Haupt- oder Unterprogramm.

Der Befehl

1. läßt das Programm zu der hinter **GOSUB** angegebenen **BASIC**-Zeile springen und beginnt dort mit dem Abarbeiten des Unterprogramms,

2. setzt das aufrufende Programm an der verlassenen Stelle fort, sobald das Unterprogramm mit dem Befehl **RETURN** verlassen wird,

3. erlaubt den Aufruf eines **BASIC**-Unterprogramms durch ein anderes **BASIC**-Unterprogramm ("NESTING").

Ist ein Unterprogramm nicht über den Befehl **RETURN** verlassen worden, muss jedes fehlende **RETURN** im weiteren Programmverlauf durch den Befehl **POP** ersetzt werden.
BASIC-Unterprogramme benutzen dieselben Variablen wie das Hauptprogramm (auch zur Datenübergabe).

(siehe Befehle **ON/GOSUB/RETURN** und **POP**)

GOSUB (GOS.)/RETURN (RET.)

1.

```
100  GOSUB 1000
200  END
1000 PRINT "UNTERPROGRAMM"
1010 RETURN
RUN
```

```
UNTERPROGRAMM
READY
```

2.

```
100  PRINT "ANFANG   ";
200  GOSUB 1000
300  PRINT "ENDE"
400  END
1000 PRINT "UNTERPR.   ";
1010 RETURN
RUN
```

```
ANFANG   UNTERPR.   ENDE
READY
```

3.

```
100  GOSUB 1000
200  PRINT "ENDE"
300  END
1000 PRINT "UNTERPR.1   ";
1010 GOSUB 2000
1020 RETURN
2000 PRINT "UNTERPR.2   ";
2010 RETURN
```

```
UNTERPR.1   UNTERPR.2   ENDE
```

ERKLÄRUNG

GOTO (G.)

Zum Beginn oder zur Fortsetzung der Programmdurchführung an der hinter **GOTO** angegebenen Stelle ("unbedingter Sprung").

Der Befehl

1. beginnt das Programm mit dem ersten Statement der **BASIC**-Zeile, die bei Dateneingabe hinter **GOTO** benannt wird (siehe Befehl **RUN**),

2. setzt das Programm mit dem ersten Statement der **BASIC**-Zeile fort, die hinter **GOTO** benannt wird,

3. läßt als Zeilenangabe Zahlen, numerische Variablen oder einen zu berechnenden Wert zu,

4. verhindert die Ausführung weiterer Statements, die hinter **GOTO** in derselben **BASIC**-Zeile stehen,

5. führt in eine endlose Schleife, wenn die Zeilennummer des Sprungzieles nicht größer ist als die Zeilennummer der Zeile, in der **GOTO** steht,

6. kann programmiert oder direkt eingegeben werden,

7. führt zu Fehler 12 und Abbruch des Programms, wenn zu einer nicht vorhandenen Zeile gesprungen wird (siehe **ERROR**).

BEISPIELE

GOTO (G.)

	Programm	Ausgabe
1.	100 PRINT "OTTO"	
	200 PRINT "EMIL"	
	GOTO 100	OTTO
		EMIL
		READY
2.	50 GOTO 200	
	100 PRINT "OTTO"	
	200 PRINT "EMIL"	EMIL
		READY
3a.	100 GOTO 1000	
3b.	100 ZEILE = 50 : GOTO ZEILE	
3c.	100 GOTO X+5	
4.	100 GOTO 200 : STOP	
	200 PRINT "ENDE"	ENDE
		READY
5.	100 PRINT "WEITER " ; : GOTO 100	WEITER WEITER WEITER WEITER . . .
6a.	100 GOTO ZEILE	
6b.	GOTO ZEILE	

ERKLÄRUNG

GRAPHICS (GR.)

Zum Auswählen eines der 12 Grafikmodi (nur PAL-Version).

Der Befehl

1. schaltet den Computer en einen der 12 Grafikmodi (siehe **GRAPHICS 0** bis **GRAPHICS 11**),
2. erwartet dazu eine Zahl von 0 bis 11 hinter **GRAPHICS,**
3. eröffnet automatisch den Bildschirm als Datenkanal #6 (entsprechend OPEN#6 , 8 , 1-11 , "S : "),
4. löscht den Bildschirm,
5. kann ohne Bildschirmlöschen verwendet werden (**GR.** ...+**32**),
6. teilt den Bildschirm in den Grafikmodi 1 bis 8 in Grafikfenster (**PRINT#6** oder **COLOR** und **PLOT/DRAWTO**) und 4-zeiliges Textfenster (**PRINT**),
7. kann ohne Textfenster verwendet werden (**GR.** ...+**16**),
8. kann im Grafikmodus 0 mit Textfenster verwendet werden (**POKE703** , **4**),
9. erlaubt die gleichzeitige Verwendung verschiedener Grafikmodi auf demselben Bildschirm; dazu wird die "Display-List" geändert,
10. **GRAPHICS 0** wird automatisch eingeschaltet nach dem
    a. Einschalten des Computers,
    b. Drücken der Taste [SYSTEM RESET],
    c. Beenden der Programmdurchführung in den Grafikmodi ohne Textfenster,
    d. Abbruch des Programmes mit Fehlermeldung.

GRAPHICS (GR.)

100 GRAPHICS 2	Grafikmodus 2 (Text)
	Ausgaben an Grafikfenster
200 PRINT#6 ; "OTTO"	
300 COLOR 65+32 : PLOT 2 , 6	
400 PRINT"OK"	Ausgabe an Textfenster
100 GR. 17	Grafik 1 ohne Textfenster
100 GRAPHICS 0	Grafikmodus 0 (Text)
200 POKE 703 , 4	Textfenster
300 POKE 752 , 1	Cursor aus
	Ausgaben an Grafikfenster
400 PRINT#6 ; "GRAFIK"	links oben
500 COLOR 65+128	Inv., großes A
600 PLOT 2 , 6	in Spalte 2, Zeile 6
	Ausgabe an Textfenster
700 PRINT"TEXT"	erste Zeile (4. von unten)
	alle Ausgaben in hellblauer Schrift auf dunkelblauem Hintergrund.

GRAPHICS 0 (GR. 0)

Grafikmodus 0 (Textmodus).
Wird automatisch eingeschaltet beim Einschalten des Computers, nach dem Drücken der Taste [SYSTEM RESET], nach der Beendigung der Programmdurchführung in den Grafikmodi ohne Textfenster nach dem Abbruch des Programmes und wird u.a. bei der Programmeingabe verwendet.
Bildschirmaufteilung:

1. 40 Spalten je Zeile (0-39),
   a. Anfangswerte von Spalte 2 bis 39,
   b. zu verändern durch **POKE 82,** links bzw. **POKE 83,** rechts (linker und rechter Rand),

2a. 24 Zeilen Textfenster (Anfangswert) (0-23),

2b. 20 Zeilen Grafikfenster plus 4 Zeilen Textfenster (**POKE 703 , 4**),

3. Ausgabe aller Zeichen des ATASCII-Codes mit den Befehlen:
   a. **PRINT,**
   b. **COLOR** (ATASCII-Code) und **PLOT/DRAWTO (POSITION) (GR.** 0),
4. 2 Farben,
   a. Fenster (Buchstaben und Hintergrund haben dieselbe Farbe),
   b. Rand (Farbe aus **SETCOLOR 4**),
5. 8 Helligkeitsstufen je Farbe,
6. Speicherbedarf für den Bildschirm: 993 Bytes.
7. die Darstellungsart von **GR. 0** gilt für alle Textfenster für
   a. Zeichenvorrat und -größe,
   b. Schrifthelligkeit aus **SETCOLOR 1,**
   c. Schrift- und Hintergrundfarbe aus **SETCOLOR** 2,

GRAPHICS 0 (GR. 0)

Anfangs-farben	SETCOLOR (Farbeimer)	COLOR (Pinsel)	Grafikfenster COLOR + PLOT	Textfenster (POS.) + PRINT
	0		–	
hellgrün	1	ATASCII	Schrifthelligkeit	
dunkelblau	2	ATASCII	Schrift- und Hintergrundfarbe	
	3		–	
schwarz	4		Randfarbe	

Der ATASCII-Code hinter COLOR bestimmt, welches Zeichen ausgegeben wird.

3a.	100 GRAPHICS 0	löscht den Bildschirm
	200 SETCOLOR 2 , 0 , 0	Hintergrundfarbe schwarz
	300 SETCOLOR 1 , 0 , 12	Zeichenhelligkeit (weiß)
	400 PRINT"HALLO"	Text weiß auf schwarz
3b.	100 GRAPHICS 0	Textmodus 0, hier notwendig.
	200 POKE 752 , 1	Cursor aus,
	300 COLOR 65 : REM GROSSES A	blaues, gr0ßes A
	400 PLOT 2 , 0 : DRAWTO 5 , 0	zeichnet Linie bis
	500 PRINT"OK"	OK
	600 GOTO 600	AAAOK

ERKLÄRUNG

GRAPHICS 1 (GR. 1)

Grafikmodus 1 (Textmodus).
Die Zeichen sind doppelt so breit wie bei Modus 0 und haben dieselbe Höhe.

Bildschirmaufteilung:

1. 20 Spalten je Zeile (0-19),
2a. 20 Zeilen Grafikfenster (0-19)
    a. plus 4 Zeilen Textfenster (**GR. 1**) oder
    b. plus 4 Zeilen Textfenster, ohne Bildschirmlöschen (**GR. 1+32**),
2b. 24 Zeilen Grafikfenster (0-23),
    a. ohne Textfenster (**GR. 1+16**) oder
    b. ohne Textfenster, ohne Bildschirmlöschen (**GR. 1+16+32**),
3. Ausgabe von Großbuchstaben und Ziffern an
    a. Grafikfenster mit den Befehlen **PRINT#6** oder **COLOR** (ATASCII-Code) und **PLOT** bzw. **DRAWTO (POSITION)**,
    b. Textfenster mit dem Befehl **PRINT**,
4. 5 Farben (Ausgangsfarben siehe Tabelle),
    a. 4 Farben + Hintergrundfarbe (SETCOLOR 0-3 + SETCOLOR 4),
    b. Textfensterfarbe (SETCOLOR 2, . .),
5. 8 Helligkeitsstufen je Farbe,
6. Umschalten auf Kleinbuchstaben + Grafikzeichen (2. Hälfte des Zeichensatzes) durch **POKE 756,226** (zurück durch **POKE 756,224**).
7. Speicherbedarf für den Bildschirm: 513 Bytes.

BEISPIELE

GRAPHICS 1 (GR. 1)

Anfangs-farben	SETCOLOR (Farbeimer)	COLOR (Pinsel)	Grafikfenster PRINT #6	Textfenster PRINT
orange	0	ATASCII	Großbuchstaben/Ziffern	–
hellgrün	1	ATASCII	Kleinbuchstaben	Schrifthelligkeit
dunkelblau	2	ATASCII	inv. Großb./Ziffern	Schrift- und Hintergrundfarbe
rot	3	ATASCII	inv. Kleinbuchstaben	–
schwarz	4		Hintergrundfarbe	–

Der ATASCII-Code hinter **COLOR** bestimmt, welches Zeichen ausgegeben wird.

```
100   GRAPHICS 1
200   COLOR 65 : REM GROSSES A
300   PLOT 0 , 5
400   PRINT"OK"
500   PRINT #6 ; "tari"
```

Textmodus 1, mit Grafikfenster
orangefarbenes, großes A,
in Spalte 0 und Zeile 5
Ausgabe im Textfenster mit hellblauen Zeichen auf dunkelblauem Hintergrund
OK
Ausgabe im Grafikfenster hinter dem A in hellgrün

6. POKE 756,226

Text erscheint nun in kleiner Schrift, Leerzeichen werden durch Herzen ersetzt.

ERKLÄRUNG

GRAPHICS 2 (GR. 2)

Grafikmodus 2 (Textmodus).
Die Zeichen sind doppelt so breit wie bei Modus 0 und haben die doppelte Höhe.
Bildschirmaufteilung:

1. 20 Spalte je Zeile (0-19),

2a. 20 Zeilen Grafikfenster (0-19),
  a. plus 4 Zeilen Textfenster (**GR. 2**) oder
  b. plus 4 Zeilen Textfenster, ohne Bildschirmlöschen (**GR. 2+32**),

2b. 24 Zeilen Grafikfenster (0-23),
  a. ohne Textfenster (**GR. 2+16**) oder
  b. ohne Textfenster, ohne Bildschirmlöschen (**GR. 2+16+32**),

3. Ausgaben von Großbuchstaben und Ziffern an
  a. Grafikfenster mit den Befehlen **PRINT #6** oder**COLOR** (ATASCII-Code) und **PLOT** bzw. **DRAWTO (POSITION),**
  b. Textfenster mit dem Befehl **PRINT,**

4. 5 Farben (Ausgangsfarben siehe Tabelle)
  a. 4 Farben + Hintergrundfarbe (**SETCOLOR 0-3** + **SETCOLOR 4**b. Textfensterfarbe (**SETCOLOR 2 , ...**)

5. 8 Helligkeitsstufen je Farbe,

6. Umschalten auf Kleinbuchstaben + Grafikzeichen (2. Hälfte des Zeichensatzes) durch **POKE 756,226** (zurück durch **POKE 756,224**) ,

7. Speicherbedarf für den Bildschirm: 261 Bytes.

GRAPHICS 2 (GR. 2)

Anfangs-farben	SETCOLOR (Farbeimer)	COLOR (Pinsel)	Grafikfenster PRINT #6	Textfenster PRINT
orange	0	ATASCII	Großb./Ziffern	–
hellgrün	1	ATASCII	Kleinbuchstaben	Schrifthelligkeit
dunkelblau	2	ATASCII	inv. Großb./Ziffern	Schrift- und Hintergrundfarbe
rot	3	ATASCII	inv. Kleibuchstaben	–
schwarz	4		Hintergrundfarbe	–

Der ATASCII-Code hinter **COLOR** bestimmt, welches Zeichen ausgegeben wird.

100 GRAPHICS 2	Textmodus 2 mit Grafikfenster
200 COLOR 65 : REM GROSSES A	orangefarbenes großes A
300 PLOT 0 , 5	in Spalte 0 und Zeile 5
400 PRINT"OK"	Ausgabe im Textfenster mit hellblauen Zeichen auf dunkelblauem Hintergrund OK
500 PRINT #6 ; "tari"	Ausgabe im Grafikfenster hinter dem A in hellgrün

6. POKE 756 , 226 — Text erscheint nun in kleiner Schrift. Leerzeichen werden durch Herzen ersetzt.

GRAPHICS 3 (GR. 3)

Grafikmodus 3.
Die Grafikpunkte sind so groß wie der Cursor im Modus 0.

Bildschirmaufteilung:

1. 40 Spalten je Zeile (0-39),
2a. 20 Zeilen Grafikfenster (0-19)
    a. plus 4 Zeilen Textfenster (**GR. 3**),
    b. plus 4 Zeilen Textfenster, ohne Bildschirmlöschen (**GR. 3+32**),
2b. 24 Zeilen Grafikfenster (0-23),
    a. ohne Textfenster (**GR. 3+16**) oder
    b. ohne Textfenster, ohne Bildschirmlöschen (**GR. 3+16+32**),
3. Ausgaben an
    a. Grafikfenster mit den Befehlen **COLOR** und **PLOT** bzw. **DRAWTO** (farbige Grafikpunkte),
    b. Textfenster mit dem Befehl **PRINT** (wie **GR. 0**),
4. 4 Farben,
    a. Grafikfenster 3 Farben,
    b. Hintergrundfarbe,
5. 8 Helligkeitsstufen je Farbe,
6. Speicherbedarf für den Bildschirm: 279 Bytes.

GRAPHICS 3 (GR. 3)

Anfangs-farben	SETCOLOR (Farbeimer)	COLOR (Pinsel)	Grafikfenster PLOT + DRAWTO	Textfenster PRINT
orange	0	1	Grafikp./CO. 1	–
hellgrün	1	2	Grafikp./CO. 2	Schrifthelligkeit
dunkelblau	2	3	Grafikp./CO. 3	Schrift- und Hintergrundfarbe
	3			
schwarz	4	0	Hintergrundfarbe	

100	GRAPHICS 3	Grafikmodus 3, mit Textfenster
200	SETCOLOR 0 , 1 , 4	Farbeimer 0 mit gold füllen
300	SETCOLOR 1 , 12 , 4	Farbeimer 1 mit grün füllen
400	SETCOLOR 2 , 8 , 10	Farbeimer 2 mit blau füllen
500	COLOR 1 : REM GOLD	Pinsel 1 mit Eimer 0 (gold)
600	PLOT 0 , 0 : DRAWTO 10 , 0	zeichnet diese Linie
700	COLOR 2 : REM GRUEN	Pinsel 2 mit Eimer 1 (grün)
800	PLOT 10 , 10 : DRAWTO 0 , 10	zeichnet diese Linie
900	COLOR 3 : REM BLAU	Pinsel 3 mit Eimer 2 (blau)
1000	DRAWTO 0 , 0	zeichnet diese Linie
1020	PRINT"OK"	Ausgabe im Textfenster mit Farbe 2 und Helligkeit von 1

GRAPHICS 4 (GR. 4)

Grafikmodus 4.
Die Grafikpunkte sind halb so breit und halb so hoch wie der Cursor im Modus 0.

Bildschirmaufteilung:

1. 80 Spalten je Zeile (0-79),
2a. 40 Zeilen Grafikfenster (0-39)
    a. plus 4 Zeilen Textfenster (**GR. 4**),
    b. plus 4 Zeilen Textfenster, ohne Bildschirmlöschen (**GR. 4+32**),
2b. 48 Zeilen Grafikfenster (0-47),
    a. ohne Textfenster (**GR. 4+16**) oder
    b. ohne Textfenster, ohne Bildschirmlöschen (**GR. 4+16+32**),
3. Ausgaben an
    a. Grafikfenster mit den Befehlen **COLOR** und **PLOT** bzw. **DRAWTO** (farbige Grafikpunkte),
    b. Textfenster mit dem Befehl **PRINT** (wie **GR. 0**),
4. 2 Farben,
    a. Grafikfenster 1 Farbe,
    b. Hintergrundfarbe,
5. 8 Helligkeitsstufen je Farbe,
6. Speicherbedarf für den Bildschirm: 537 Bytes.

GRAPHICS 4 (GR. 4)

Anfangs-farben	SETCOLOR (Farbeimer)	COLOR (Pinsel)	Grafikfenster PLOT + DRAWTO	Textfenster PRINT
orange	0	1	Grafikp./CO. 1	–
hellgrün	1	2		Schrifthelligkeit
dunkelblau	2	3		Schrift- und Hintergrundfarbe
	3			
schwarz	4	0	Hintergrundfarbe	

100 GRAPHICS 4	Grafikmodus 4, mit Textfenster
200 SETCOLOR 0 , 1 , 4	Farbeimer 0 mit gold füllen
300 COLOR 1 : REM GOLD	Pinsel 1 mit Eimer 0 (gold)
400 PLOT 0 , 0 : DRAWTO 10 , 0	zeichnet diese Linie
500 PLOT 10 , 10 : DRAWTO 0 , 10	zeichnet diese Linie
600 COLOR 3 : REM DUNKELBLAU	Pinsel 3 mit Eimer 2 (blau)
700 DRAWTO 0 , 0	zeichnet diese Linie
800 PRINT"OK"	Ausgabe im Textfenster mit Farbe von Eimer 2 und Helligkeit von Eimer 1 OK

GRAPHICS 5 (GR. 5)

Grafikmodus 5.
Die Grafikpunkte sind halb so breit und halb so hoch wie der Cursor im Modus 0.

Bildschirmaufteilung:

1. 80 Spalten je Zeile (0-79),
2a. 40 Zeilen Grafikfenster (0-39)
    a. plus 4 Zeilen Textfenster (**GR. 5**),
    b. plus 4 Zeilen Textfenster, ohne Bildschirmlöschen (**GR. 5+32**),
2b. 48 Zeilen Grafikfenster (0-47),
    a. ohne Textfenster (**GR. 5+16**) oder
    b. ohne Textfenster, ohne Bildschirmlöschen (**GR. 5+16+32**),
3. Ausgaben an
    a. Grafikfenster mit den Befehlen **COLOR** und **PLOT** bzw. **DRAWTO** (farbige Grafikpunkte),
    b. Textfenster mit dem Befehl **PRINT** (wie **GR. 0**),
4. 4 Farben,
    a. Grafikfenster 3 Farben,
    b. Hintergrundfarbe,
5. 8 Helligkeitsstufen je Farbe,
6. Speicherbedarf für den Bildschirm: 1017 Bytes.

GRAPHICS 5 (GR. 5)

Anfangs-farben	SETCOLOR (Farbeimer)	COLOR (Pinsel)	Grafikfenster PLOT + DRAWTO	Textfenster PRINT
orange	0	1	Grafikp./CO. 1	–
hellgrün	1	2	Grafikp./CO. 2	Schrifthelligkeit
dunkelblau	2	3	Grafikp./CO. 3	Schrift- und Hintergrundfarbe
	3			
schwarz	4	0	Hintergrundfarbe	

```
 100  GRAPHICS 5                       Grafikmodus 5, mit Textfenster
 200  SETCOLOR 0 , 1 , 4               Farbeimer 0 mit gold füllen
 300  SETCOLOR 1 , 12 , 4              Farbeimer 1 mit grün füllen
 400  SETCOLOR 2 , 8 , 10              Farbeimer 2 mit blau füllen
 500  COLOR 1 : REM GOLD               Pinsel 1 mit Eimer 0 (gold)
 600  PLOT 0 , 0 : DRAWTO 10 , 0       zeichnet diese Linie
 700  COLOR 2 : REM GRUEN              Pinsel 2 mit Eimer 1 (grün)
 800  PLOT 10 , 10 : DRAWTO 0 , 10     zeichnet diese Linie
 900  COLOR 3 : REM BLAU               Pinsel 3 mit Eimer 2 (blau)
1000  DRAWTO 0 , 0                     zeichnet diese Linie
1020  PRINT"OK"                        Ausgabe im Textfenster
```

GRAPHICS 6 (GR. 6)

Grafikmodus 6.
Die Grafikpunkte sind zwei Fernsehzeilen hoch und ebenso breit.

Bildschirmaufteilung:

1. 160 Spalten je Zeile (0-159),
2a. 80 Zeilen Grafikfenster (0-79)
    a. plus 4 Zeilen Textfenster (**GR. 6**),
    b. plus 4 Zeilen Textfenster, ohne Bildschirmlöschen (**GR. 6+32**),
2b. 96 Zeilen Grafikfenster (0-95),
    a. ohne Textfenster (**GR. 6+16**) oder
    b. ohne Textfenster, ohne Bildschirmlöschen (**GR. 6+16+32**),
3. Ausgaben an
    a. Grafikfenster mit den Befehlen **COLOR** und **PLOT** bzw. **DRAWTO** (farbige Grafikpunkte),
    b. Textfenster mit dem Befehl **PRINT** (wie **GR. 0**),
4. 2 Farben,
    a. Grafikfenster 1 Farbe,
    b. Hintergrundfarbe,
5. 8 Helligkeitsstufen je Farbe,
6. Speicherbedarf für den Bildschirm: 2025 Bytes.

GRAPHICS 6 (GR. 6)

Anfangs-farben	SETCOLOR (Farbeimer)	COLOR (Pinsel)	Grafikfenster PLOT + DRAWTO	Textfenster PRINT
orange	0	1	Grafikp./CO. 1	–
hellgrün	1	2		Schrifthelligkeit
dunkelblau	2	3		Schrift- und Hintergrundfarbe
	3			
schwarz	4	0	Hintergrundfarbe	

```
100 GRAPHICS 6                      Grafikmodus 6, mit Textfenster
200 SETCOLOR 0 , 1 , 4              Farbeimer 0 mit gold füllen
300 COLOR 1 : REM GOLD              Pinsel 1 mit Eimer 0 (gold)
400 PLOT 0 , 0 : DRAWTO 10 , 0      zeichnet diese Linie
500 PLOT 10 , 10 : DRAWTO 0 , 10    zeichnet diese Linie
600 COLOR 3 : REM DUNKELBLAU        Pinsel 3 mit Eimer 2 (blau)
700 DRAWTO 0 , 0                    zeichnet diese Linie
800 PRINT"OK"                       Ausgabe im Textfenster mit Farbe von Eimer 2 und Helligkeit
                                    von Eimer 1
                                    OK
```

GRAPHICS 7 (GR. 7)

Grafikmodus 7.
Die Grafikpunkte sind zwei Fernsehzeilen hoch und ebenso breit.

Bildschirmaufteilung:

1. 160 Spalten je Zeile (0-159),
2a. 80 Zeilen Grafikfenster (0-79)
    a. plus 4 Zeilen Textfenster (**GR. 7**),
    b. plus 4 Zeilen Textfenster, ohne Bildschirmlöschen (**GR. 7+32**),
2b. 96 Zeilen Grafikfenster (0-95),
    a. ohne Textfenster (**GR. 7+16**) oder
    b. ohne Textfenster, ohne Bildschirmlöschen (**GR. 7+16+32**),
3. Ausgaben an
    a. Grafikfenster mit den Befehlen **COLOR** und **PLOT** bzw. **DRAWTO** (farbige Grafikpunkte),
    b. Textfenster mit dem Befehl **PRINT** (wie **GR. 0**),
4. 4 Farben,
    a. Grafikfenster 3 Farben,
    b. Hintergrundfarbe,
5. 8 Helligkeitsstufen je Farbe,
6. Speicherbedarf für den Bildschirm: 3945 Bytes.

BEISPIELE

GRAPHICS 7 (GR. 7)

Anfangs-farben	SETCOLOR (Farbeimer)	COLOR (Pinsel)	Grafikfenster PLOT + DRAWTO	Textfenster PRINT
orange	0	1	Grafikp./CO. 1	–
hellgrün	1	2	Grafikp./CO. 2	Schrifthelligkeit
dunkelblau	2	3	Grafikp./CO. 3	Schrift- und Hintergrundfarbe
	3			
schwarz	4	0	Hintergrundfarbe	

```
100   GRAPHICS 7                     Grafikmodus 7, mit Textfenster
200   SETCOLOR 0 , 1 , 4             Farbeimer 0 mit gold füllen
300   SETCOLOR 1 , 12 , 4            Farbeimer 1 mit grün füllen
400   SETCOLOR 2 , 8 , 10            Farbeimer 2 mit blau füllen
500   COLOR 1 : REM GOLD             Pinsel 1 mit Eimer 0 (gold)
600   PLOT 0 , 0 : DRAWTO 10 , 0     zeichnet diese Linie
700   COLOR 2 : REM GRUEN            Pinsel 2 mit Eimer 1 (grün)
800   PLOT 10 , 10 : DRAWTO 0 , 10   zeichnet diese Linie
900   COLOR 3 : REM BLAU             Pinsel 3 mit Eimer 2 (blau)
1000  DRAWTO 0 , 0                   zeichnet diese Linie
1020  PRINT"OK"                      Ausgabe im Textfenster
```

ERKLÄRUNG

GRAPHICS 8 (GR. 8)

Grafikmodus 8.
Die Grafikpunkte sind eine Fernsehzeile hoch und ebenso breit.

Bildschirmaufteilung:

1. 320 Spalten je Zeile (0-319),
2a. 160 Zeilen Grafikfenster (0-159)
    a. plus 4 Zeilen Textfenster (**GR. 8**),
    b. plus 4 Zeilen Textfenster, ohne Bildschirmlöschen (**GR. 8+32**),
2b. 192 Zeilen Grafikfenster (0-191),
    a. ohne Textfenster (**GR. 8+16**) oder
    b. ohne Textfenster, ohne Bildschirmlöschen (**GR. 8+16+32**),
3. Ausgaben an
    a. Grafikfenster mit den Befehlen **COLOR** und **PLOT** bzw. **DRAWTO** (farbige Grafikpunkte),
    b. Textfenster mit dem Befehl **PRINT** (wie **GR. 0**),
4. 2 Farben,
    a. Grafikfenster 1 Farbe, Grafikpunkt und Hintergrund haben dieselbe Farbe, aber verschiedene Helligkeit,
    b. Randfarbe,
5. 8 Helligkeitsstufen je Farbe,
6. Speicherbedarf für den Bildschirm: 7900 Bytes.

GRAPHICS 8 (GR. 8)

Anfangs-farben	SETCOLOR (Farbeimer)	COLOR (Pinsel)	Grafikfenster PLOT + DRAWTO	Textfenster PRINT
	0			–
hellgrün	1	1	Grafikpunkthelligkeit	Schrifthelligkeit
dunkelblau	2	0	Grafikpunkthintergrund	Schrift- und Hintergrundfarbe
	3			
schwarz	4		Randfarbe	

100 GRAPHICS 8	Grafikmodus 8, mit Textfenster
200 SETCOLOR 1 , 12 , 10 : REM HELLIGKEIT	Farbeimer 1 mit Helligkeit 10
300 SETCOLOR 2 , 4 , 4 : REM ROT	Farbeimer 2 mit rot füllen
400 SETCOLOR 4 , 10 , 6 : REM GRUENBLAU	Farbeimer 4 mit grünblau füllen
500 COLOR 1 : REM HELLIGKEIT	Pinsel 1 mit Helligkeit von Eimer 1 (hell)
600 PLOT 0 , 0 : DRAWTO 319 , 159	zeichnet diese Linie
700 PRINT"OK"	Ausgabe im Textfenster mit Farbe von Eimer 2 und Helligkeit von Eimer 1 OK

ERKLÄRUNG

GRAPHICS 9 (GR. 9)

Grafikmodus 9.

Die Grafikpunkte sind so hoch wie eine Fernsehzeile und 4 mal so breit.

Bildschirmaufteilung:

1. 80 Spalten je Zeile (0-79),

2. 192 Zeilen Grafikfenster (0-191), kein Textfenster möglich,

3. Ausgaben an
   Grafikfenster mit den Befehlen **COLOR** und **PLOT** bzw. **DRAWTO** (farbige Grafikpunkte),

4. 1 Farbe,
   a. Grafikfenster,
   b. Rand- und Hintergrundhelligkeit von **SE. 4,**

5. 16 Helligkeitsstufen je Farbe,

6. Speicherbedarf für den Bildschirm: 7900 Bytes.

   Hinweis: Die Farbe von **SE. 4** bestimmt die Farbe in allen Helligkeiten.

GRAPHICS 9 (GR. 9)

Anfangs-farben	SETCOLOR (Farbeimer)	COLOR (Pinsel)	Grafikfenster PLOT + DRAWTO
schwarz	4	0	Grafikpunkte/Hintergrundfarbe
	,	1	Grafikpunkt/Helligkeit 1
		2	Grafikpunkt/Helligkeit 2
		3	Grafikpunkt/Helligkeit 3
		4	Grafikpunkt/Helligkeit 4
		5 – 14	Helligkeiten 5 – 14
		15	Grafikpunkt/Helligkeit 15

```
100  GRAPHICS 9                            Grafikmodus 9, ohne Textfenster
200  SETCOLOR 4 , 3 , 0 : REM DUNKELROT    Farbeimer 4 füllen mit rot
300  FOR X=0 TO 15                         Schleife für alle
400  COLOR X                               Helligkeiten
500  PLOT X , 0 : DRAWTO X , 191           senkrechte Balken mit steigender Helligkeit
600  NEXT X
700  GOTO 700
```

GRAPHICS 10 (GR. 10)

Grafikmodus 10.

Die Grafikpunkte sind so hoch wie eine Fernsehzeile und 4 mal so breit.

Bildschirmaufteilung:

1. 80 Spalten je Zeile (0-79),
2. 192 Zeilen Grafikfenster (0-191), kein Textfenster möglich,
3. Ausgaben an Grafikfenster mit den Befehlen **COLOR** und **PLOT** bzw. **DRAWTO** (farbige Grafikpunkte),
4. 9 Farben,
   a. Grafikfenster 9 Farben,
   b. Rand- und Hintergrundfarbe ist eine davon,
5. 8 Helligkeitsstufen je Farbe,
6. Speicherbedarf für den Bildschirm: 7900 Bytes.

   Hinweis: Die Farben, die hier **COLOR 0** bis **3** zugeordnet sind, werden mit der folgenden Formel gesetzt:
   POKE Farbadresse , Farbkennzahl*16+Helligkeitsstufe.

GRAPHICS 10 (GR. 10)

Anfangs- farben	SETCOLOR/POKE (Farbeimer)		COLOR (Pinsel)	Grafikfenster PLOT + DRAWTO
schwarz		704	0	Grafikpunkte/Hintergrundfarbe
schwarz		705	1	Grafikpunkt
schwarz		706	2	Grafikpunkt
schwarz		707	3	Grafikpunkt
orange	0	708	4	Grafikpunkt
hellgrün	1	709	5	Grafikpunkt
dunkelblau	2	710	6	Grafikpunkt
rot	3	711	7	Grafikpunkt
schwarz	4	712	8	Grafikpunkt

```
100  GRAPHICS 10                           Grafikmodus 10, ohne Textfenster
200  POKE 705 , 3*16+10 : REM ROT , HELL   Farbe für Pinsel 1
300  SETCOLOR 2 , 9 , 2 : REM BLAU , DUNKEL   Farbeimer 2 mit blau füllen
400  FOR X=0 TO 8                          Schleife für alle
500  COLOR X                               Farben
600  PLOT X , 0 : DRAWTO X , 191           senkrechte Balken zeichnen
700  NEXT X
800  GOTO 800
```

ERKLÄRUNG

GRAPHICS 11 (GR. 11)

Grafikmodus 11.

Die Grafikpunkte sind so hoch wie eine Fernsehzeile und 4 mal so breit.

Bildschirmaufteilung:

1. 80 Spalten je Zeile (0-79),
2. 192 Zeilen Grafikfenster (0-191), kein Textfenster möglich,
3. Ausgaben an Grafikfenster mit den Befehlen **COLOR** und **PLOT** bzw. **DRAWTO** (farbige Grafikpunkte),
4. 16 Farben,
   a. Grafikfenster 16 Farben,
   b. Rand- und Hintergrundhelligkeit ist eine davon,
5. 1 gemeinsame Helligkeit für alle Farben,
6. Speicherbedarf für den Bildschirm: 7900 Bytes.

   Hinweis: Die Helligkeit von **SE. 4** bestimmt die Farbe in allen Farben.

GRAPHICS 11 (GR. 11)

Anfangs- farben	SETCOLOR (Farbeimer)	COLOR (Pinsel)	Grafikfenster PLOT + DRAWTO
schwarz	4	0	Grafikpunkte/Hintergrundfarbe
gold		1	Grafikpunkt
orange		2	Grafikpunkt
rot		3	Grafikpunkt
rot		4	Grafikpunkt
lila		5	Grafikpunkt
blau		6 – 9	Grafikpunkt
türkis		10	Grafikpunkt
grün		11 – 14	Grafikpunkt
hellorange		15	Grafikpunkt

```
100  GRAPHICS 11                      Grafikmodus 11, ohne Textfenster
200  SETCOLOR 4 , 0 , 6 : REM DUNKEL  Farbeimer 4 mit dunkel füllen
300  FOR X=0 TO 15                    Schleife für alle
400  COLOR X                          Farben
500  PLOT X , 0 : DRAWTO X , 191      senkrechte Balken zeichnen
600  NEXT X
700  GOTO 700
```

ERKLÄRUNG

IF/THEN

Zur Fortsetzung des Programms an der hinter **THEN** bezeichneten Stelle, wenn die gestellte Bedingung (zwischen **IF** und **THEN**) erfüllt ist ("bedingter Sprung").

Der Befehl

1. prüft Bedingungen
   a) mit Strings,
   b) mit Zahlen,
   c) mit boolschen Ausdrücken (wahr=1 und falsch=0),
   d) mit zu berechnenden Werten
   und
2. führt bei erfüllter Bedingung zur Ausführung der nach **THEN** stehenden Statements,
3. führt bei nicht erfüllter Bedingung zur nächsten **BASIC**-Zeile,
4. läßt als Statements zu
   a) eine Zeilennummer als Sprungziel,
   b) ein weiteres **IF/THEN** in derselben Zeile,
   c) weitere Statements in derselben Zeile,
5. läßt als Sprungziel eine numerische Variable nur mit zusätzlichem **GOTO** oder **GOSUB** zu (die Variable wird bei Bedarf gerundet).

BEISPIELE

IF/THEN

```
1a.  100  IF A$ = "Otto" THEN 200
1b.  100  IF A = 10 THEN 300
1c.  100  A = 1 : B = 1 : IF A AND B THEN ? "OK"
1d.  100  B = 1 : C = 0 : D = 0 : E = 1
     200  IF A = B+C OR D<E THEN 400
     300  STOP
     400  STOP
2.   100  A = 10 : IF A = 10 THEN 300
     200  END
     300  PRINT A
3.   100  A = 9 : IF A = 10 THEN 300
     200  END
     300  PRINT A
4a.  100  IF A = 0 THEN 1000
4b.  100  IF A = 0 THEN IF Y = 3 THEN 1000
4c.  100  IF A = 0 THEN PRINT A : PRINT C
5.   100  ZEILE = 400 : A = 10
     200  IF A = 10 THEN GOTO ZEILE
     300  STOP
     400  ? "OK"
```

INDIZIERTE VARIABLEN

IM **ATARI-BASIC** sind indizierte Variablen immer numerische Variablen (siehe numerische Variablen).

1. Der Name einer einfach indizierten Variablen enthält am Ende einen in Klammern gesetzten, eventuell noch zu berechnenden Wert (den Index), der den Listenplatz der Variablen in einer eindimensionalen Speicherplatzliste ("ARRAY") angibt.

2. Der Name einer doppelt indizierten Variablen enthält am Ende zwei in Klammern gesetzte und durch Komma getrennte, eventuell noch zu berechnende Werte. Der jeweils Erste gibt die Zeilenposition, der Zweite die Spaltenposition der Variablen in einer zweidimensionalen Speicherplatztabelle ("MATRIX") an.

INDIZIERTE VARIABLEN

1. Zuweisung eines Wertes zu einer einfach indizierten Variablen:

```
• 100 DIM A(1) : A(1) = 2.5 : PRINT A(1)                 2.5
• 100 DIM MONAT(12) : MONAT(2) = 28
  200 PRINT MONAT(2)                                     28
• 100 ANFANG = 0 : SCHLUSS = 4 :
      DIM SPEICHER(4) : DATA 1 , 2 , 3 , 4 , 5
  200 FOR X = ANFANG TO SCHLUSS
  300 READ A : SPEICHER(X) = A : NEXT X
  400 PRINT SPEICHER(2)                                  3
```

2. Zuweisung eines Wertes zu einer doppelt indizierten Variablen:

```
• 100 DIM TAB(2,2)
  200 TAB(0,0) = 1 : TAB(0,1) = 2
  300 TAB(1,0) = 4 : TAB(1,1) = 5
  400 PRINT TAB(1,0)                                     4
• 100 B = 2 : C = 3 : DIM A(B,C)
  200 A(1,2) = 17
  300 PRINT A(1,2)                                       17
```

1.) DIM A(4) ergibt:

A	0	1	2	3	4
	A(0)	A(1)	A(2)	A(3)	A(4)

2.) DIM A(4,2) ergibt:

A	0	1	2	3	4
0	A(0,0)	A(0,1)	A(0,2)	A(0,3)	A(0,4)
1	A(1,0)	A(1,1)	A(1,2)	A(1,3)	A(1,4)
2	A(2,0)	A(2,1)	A(2,2)	A(2,3)	A(2,4)

ERKLÄRUNG

INPUT (I.)

Zur Zuweisung von Eingaben zu Variablen.

Der Befehl verarbeitet die Eingabe

1. von Text für eine Stringvariable,

2. einer Zahl für eine numerische Variable (die nicht doppelt indiziert sein darf),

3a. von mehreren, durch Kommas getrennten Zahlen für die entsprechende Anzahl von numerischen Variablen (die nicht doppelt indiziert sein dürfen).

3b. von mehreren Texten oder Zahlen für eine entsprechende Anzahl von String- bzw. numerischen Variablen,

4. von anderen Eingabegeräten als der Tastatur (mit der Nummer des Datenkanals; erkennt dabei EOL als Datensatzende).

5. der Taste [RETURN] ohne vorangehende Daten (nur für die Zuordnung zu Stringvariablen),

6. häufig wird einem INPUT-Befehl ein PRINT-Befehl vorangestellt, der die einzugebenden Daten näher beschreibt,

7. ist der **INPUT**-Befehl bei der Programmdurchführung erreicht, hält das Programm an, gibt ein ? aus und erwartet die Eingabe der Daten, die mit der Taste [RETURN] abgeschlossen wird. Danach wird die Programmausführung fortgesetzt.

BEISPIELE

INPUT (I.)

1.	100	DIM A$(4)	
	200	INPUT A$: PRINT A$	? OTTO
			OTTO
2.	100	INPUT A : PRINT A	? 10
			10
3a.	100	INPUT A , B , C : PRINT A , B , C	? 1 , 17 , 128
			1 17 128
3b.	100	PRINT "NAME, DANACH ALTER EINGEBEN:"	NAME, DANACH ALTER EINGEBEN:
	200	DIM NAME$(10) : INPUT NAME , ALTER	? OTTO
	300	PRINT NAME$; "IST" ; ALTER	? 20
			OTTO IST 20
4.	100	OPEN #1 , 4 , 0 , D :	
	200	INPUT #1 , A$: PRINT A$, : GOTO 200	
5a.	100	DIM A$(4)	
	200	INPUT A$: PRINT A$	? Taste [RETURN] (A$ ist leer)
5b.	100	INPUT A : PRINT A	? Taste [RETURN] ERROR 6 AT LINE 100
			(die Eingabe wird abgewiesen)
6.	100	PRINT"WIEVIEL " ; : INPUT A	
	200	PRINT : PRINT A ; " CM"	WIEVIEL ? 17 Taste [RETURN]
			17 CM

Taste INSERT

Zum Einschieben von Zeichen oder Zeilen.

Wird die Taste [INSERT] gleichzeitig gedrückt mit

1. der Taste [CTRL], so wird an der Cursorposition ein Leerzeichen eingeschoben; der Rest der logischen Zeile rückt eine Position nach rechts,

2. der Taste [SHIFT], so wird an der Cursorposition eine Leerzeile eingeschoben; der restliche Bildschirminhalt wird eine Zeile nach unten verschoben; dabei geht der Inhalt der Zeile, die vorher am unteren Bildschirmrand war, verloren.

BEISPIELE

Taste INSERT

1. ABCDEF — Cursor auf C
   zweimal Tasten CTRL und INSERT
   AB□ CDEF — zwei Leerzeichen eingeschoben

2. ABCDEF
   GH IJKL — Cursor auf H
   MNOPQR
   STUVWX
   ———————— Bildschirmunterkante

   zweimal Tasten SHIFT und INSERT
   ABCDEF
   □
   — zwei Leerzeilen eingeschoben
   GHIJKL
   ———————— Bildschirmunterkante

ERKLÄRUNG

INT

Zum Runden von Zahlen.

Die Funktion

1. errechnet die größte, ganze Zahl, die kleiner oder gleich dem Wert in der Klammer ist,

2. verarbeitet in der Klammer
   a. eine Zahl,
   b. eine Variable oder
   c. einen zu berechnenden Wert,

3. wird häufig verwendet
   a. zum Aufrunden,
   b. zum Abrunden,
   c. zum Festlegen einer bestimmten Anzahl von Nachkommastellen,
   d. zum Feststellen, ob eine Zahl ohne Rest durch eine Andere teilbar ist.

BEISPIELE

INT

1a.	100	PRINT INT(14.657)	14
1b.	100	PRINT INT(−15.5)	−16
2a.	100	PRINT INT(3.5)	3
2b.	100	A = 3.5 : PRINT INT(A)	3
2c.	100	PRINT INT(27−30.3)	−4
3a.	100	C=0.99999998	
	200	C=INT(C*1E5+.5)/1E5	Aufrunden an der 5. Stelle
	300	PRINT C	1
3b.	100	C=0.99999998	
	200	C=INT(C*1E5−.5)/1E5	Abrunden an der 5. Stelle
	300	PRINT C	0.99999
3c.	100	B=1.374	
	200	B=INT(B*100)/100	Festlegen auf 2 Nachkommastellen
	300	PRINT B	1.37
3d.	100	PRINT"EINGABE " ; : INPUT A	
	200	IF (A/10)=INT(A/10) THEN PRINT"OK" : GOTO 100	Abprüfen, ob A 10, 20 oder 30 usw. ist
	300	PRINT"ANDERE " ; : GOTO 100	

```
RUN
EINGABE ?9
ANDERE EINGABE ?10
OK
EINGABE ?
```

Interner Code

Versetzter ATASCII-Code, der vom Computer intern benutzt wird.

Der interne Code

1. findet beim Ablegen von ATASCII-Zeichen in Speicherstellen Anwendung,

2. wird auch im Bildschirmbereich verwendet,

3. eines Zeichens kann mit dem Befehl
   a. **PEEK** gelesen werden,
   b. **POKE** verändert werden,

4. eines Zeichens kann aus dem ATASCII-Code berechnet werden,

5. eines Zeichens kann in den ATASCII-Code umgerechnet werden.

   (siehe ATASCII-Code und Befehle **PEEK** und **POKE**)

Interner Code

1. `100 POKE 40000 , A : A=A+1 : FOR X=0 TO 50 0 : NEXT X : ?CHR$(125) ; "A= " ; A : GOTO 100` — Ausdruck des internen Codes und des entsprechenden Zeichens

2a. `100 POKE 40000 , 41` — Großes I in der linken oberen Bildschirmecke

2b. `100 PRINT PEEK(40000)` — 41
der interne Code des großen I

3. ATASCII umwandeln in internen:

0 – 31	ATASCII +64
32 – 95	ATASCII −32
96 – 127	Identisch
128 – 159	ATASCII +64
160 – 223	ATASCII −32
224 – 255	Identisch

4. Internen umwandeln in ATASCII:

0 – 63	Intern +32
64 – 95	Intern −64
96 – 127	Identisch
128 – 191	Intern +32
192 – 223	Intern −64
224 – 255	Identisch

ERKLÄRUNG

LEN

Zum Feststellen der Länge einer String-Variablen oder eines Textes.

Die Funktion

1. findet die Anzahl der Zeichen in
   a. einer String-Variablen oder
   b. einem Text,

2. ergibt 0, wenn darin keine Zeichen enthalten sind,

3. zählt dabei alle Zeichen mit, die vorkommen.

BEISPIELE

LEN

1a.	100	DIM A$(4)	
	200	A$="EMIL"	
	300	A=LEN(A$)	
	400	PRINT A	4
1B.	500	PRINT LEN("EINS")	4
2.	100	DIM A$(4) , B$(4)	
	200	B=LEN(B$)	
	300	PRINT B	0
3.	400	A$=""	
	500	PRINT LEN(A$)	0 Zum Löschen des Inhalts von A$
4.	600	PRINT LEN(" ")	2 zwei Zwischenräume

ERKLÄRUNG

LET (LE.)

Zur Festlegung von Variablen.

Der Befehl

1. definiert eine Variable,
2. weist einer Variablen einen Wert zu,
3. kann programmiert werden,
4. kann direkt eingegeben werden,
5. ist nur erforderlich, wenn der Variablenname ein geschütztes **BASIC**-Wort ist.

BEISPIELE

LET (LE.)

```
1a.  100  LET A=10
1b.  100  DIM A$(4) : LET A$= "OTTO"

2.   100  LET B=20
     500  LET B=50
     510  PRINT B

3.   100  LET C=0

4.   LET C=0

5a.  100  RUN = 100                 100 ERROR- RUN = 100

5b.  100  LET RUN = 100
     110  PRINT RUN                 100
     RUN                            READY
```

ERKLÄRUNG

LIST (L.)

Zum Schreiben eines im Speicher stehenden Programmes (in nicht übersetzter Form) auf den Bildschirm oder zu anderen Ausgabegeräten.

Der Befehl

1. gibt die **BASIC**-Zeilen eines Programmes in steigender Folge aus,
2. gibt das gesamte Programm auf dem Bildschirm aus, wenn nur **LIST** ohne Zusätze eingegeben wurde,
3. gibt den Teil des Programmes auf dem Bildschirm aus, der hinter **LIST** mit der Anfangs- und Endzeilennummer angegeben wurde,
4. gibt nur eine **BASIC**-Zeile des Programmes auf dem Bildschirm aus, wenn hinter **LIST** nur eine Zeilennummer angegeben wurde,
5. gibt ohne Zusatzangabe hinter **LIST** auf den Bildschirm aus,
6. gibt an den Drucker aus, wenn "P:" hinter **LIST** angegeben wurde,
7. gibt an die Diskette aus, wenn "D:" hinter **LIST** angegeben wurde (siehe **ENTER**),
8. gibt an die Kassette aus, wenn "C:" hinter **LIST** angegeben wurde (siehe **ENTER**),
9. läßt jede Ausgabeart an jedes Ausgabegerät zu ("S : ", "E : ", "R 1–4"),
10. wird in der Regel direkt eingegeben
11. kann auch programmiert werden

Anmerkung: Sollen nur bestimmte Zeilennummern ausgegeben werden und ist der Befehl **LIST** durch einen Ausgabekenner (z.B. "P : " oder "D : " etc.) erweitert, muß zwischen diesem und der ersten Zeilennummer ein Komma stehen z.B.:
LIST "C : " , 10 , 100.

BEISPIELE

LIST (L.)

2. LIST

3. LIST 100 , 1000

4. LIST 135

6. LIST "P:" ,100 , 1000

7a. LIST "D1 : DEMO.LST" (speichert das Programm in Textformat (ASCII))

7b.
```
100   DIM A$(15) : A$="D : DEMO.LST"
200   LIST A$
```

8. LIST "C : " ,100 , 1000 (speichert die Zeilen 100 bis 1000 in Textformat)

9. LIST "E : "

11.
```
100   LIST 100 , 1000 : PRINT "FERTIG"
```

ERKLÄRUNG

LOAD (LO.)

Zum Laden eines Programmes, in den Computer-Speicher (RAM).

Der Befehl

1. löscht ein im Speicher stehendes Programm samt Variablen,

2. lädt ein Programm in der übersetzten Form,

3. lädt nur Programme, die mit **SAVE** gespeichert worden sind,

4. kann durch den Befehl **RUN** mit Geräteangabe und Programmnamen ersetzt werden,

5. kann a) vom Kassettenrekorder oder
   b) von der Diskette lesen,

6. muß mit einem Programmnamen versehen werden, wenn von der Diskette geladen wird,

7. wird in der Regel direkt eingegeben,

8. kann auch programmiert werden.

LOAD (LO.)

zu

4a. LOAD"C : "
4b. LOAD"D : PROGRAMM.001"

5a. LOAD"C : PROGR1.BAS"
5b. LOAD"D : PROGR2.BAS"

6. LOAD"D : TEST"

7. 100 LOAD"C : "

ERKLÄRUNG

LOCATE (LOC.)

Zum Lesen eines Grafikpunktes im Grafikfenster.

Der Befehl: **LOCATE,** horizontal, vertikal, Variable

1. wirkt in allen Grafikmodi,

2a. enthält die horizontale Position an erster Stelle,

2b. enthält die vertikale Position an zweiter Stelle,

2c. enthält die Variable, in die der gelesene Wert gebracht wird,

3. findet
   a. in den Grafikmodi 0, 1 und 2 eine Zahl von 0 bis 255 (ATASCII),
   b. in den Grafikmodi 3 bis 11 den **COLOR**-Wert, mit dem dieser Grafikpunkt geschrieben worden ist,

4. entspricht **POSITION horizontal, vertikal : GET #6 , VARIABLE,**

5. setzt als Nullpunkt für die Positionsangaben die linke, obere Bildschirmecke voraus,

6. ändert in den Grafikmodi 0, sowie 9 bis 11 den Inhalt des Grafikpunktes, der gelesen worden ist, wenn der Befehl **PRINT** folgt.

LOCATE (LOC.)

```
100  GRAPHICS 0 : POSITION 0 , 0                linker oberer Rand
200  ? "ABCDEF"                                 Text ausgeben
300  FOR X=0 TO 5
400  LOC. X , 0 , A
500  POS. X , 0 : PRINT CHR$(A)                 Punkt restaurieren
600  POS. 3*X , 10 : PRINT A                    Werte ausgeben
700  POS. X , 20 : PRINT CHR$(A)                Text kopieren
800  NEXT X

100  GRAPHICS 3 : REM AUCH FUER GR. 4-7
200  COLOR 1 : PLOT 2 , 0                       Grafikpunkt orange
300  LOCATE 2 , 0 , A : PRINT A                 1   (im Textfenster)

100  GRAPHICS 8
200  SETCOLOR 1 , 4 , 10 : SETCOLOR 2 , 9 , 0   Grafikpunkt hellblau/Hintergrund dunkelblau
300  COLOR 1 : PLOT 20 , 20
400  LOCATE 20 , 20 , A : PRINT A               1

100  GRAPHICS 9 : REM AUCH FUER GR. 10-11
200  SETCOLOR 4 , 9 , 0                         Hintergrund dunkelblau
300  COLOR 14 : PLOT 20 , 20                    sehr hell
400  LOCATE 20 , 20 , A : REM PRINT A           hier versuchsweise REM entfernen
500  GOTO 500
```

ERKLÄRUNG

LOG

Zum Feststellen des natürlichen Logarithmus.

Die Funktion

1. errechnet den natürlichen Logarithmus zur Basis e (2.71828283)
   a. der Zahl,
   b. der Variablen oder
   c. des zu berechnenden Wertes

die/der in der Klammer hinter **LOG** steht,

2. erwartet einen positiven Klammerinhalt.

BEISPIELE

LOG

1a.	100	PRINT LOG(100)	4.60517018
1b.	100	C=100 : PRINT LOG(C)	4.60517018
1c.	100	PRINT LOG(200-100)	4.60517018
2a.*	100	PRINT LOG(1)	4.60517018E-10 (Rundungsfehler, eigentlich 0)
2b.*	100	PRINT LOG(0)	Abfangen, weil unzulässig, da LOG nur positiv definiert, siehe Beispiel 2c
2c.	100	PRINT LOG(ABS(-100))	4.60517018

* Auf Grund eines Fehlers im BASIC führen diese beiden Werte nicht zu den richtigen Ergebnissen.

ERKLÄRUNG

LPRINT (LP.)

Zur Ausgabe von Informationen auf dem Zeilendrucker (siehe Befehl **PRINT**).

Der Befehl

1. leitet die Ausgaben zu dem Drucker,

2. enthält alle Möglichkeiten des **PRINT**-Befehls, außer die Ausgabe erfolgt im ATASCII-Code. Dieser unterscheidet sich in einigen Fällen vom ASCII-Code. Das Resultat ist vom angeschlossenen Drucker und dessen Interpretation der Zeichen abhängig. Komma, Strichpunkt und Doppelpunkt zwischen zwei **LPRINT**-Befehlen werden analog zum **PRINT**-Befehl ausgeführt.

3. benötigt kein OPEN- oder CLOSE-Statement,

4. kann programmiert werden,

5. kann auch direkt eingegeben werden.

6. **LPRINT** ohne Text bewirkt Leerzeile (Zeichen die noch im Speicher des Druckers stehen werden sofort ausgegeben).

7. Das Vorhandensein eines Druckers wird vom Computer geprüft (siehe **ERROR:** –ERROR 138).

LPRINT (LP.)

BEISPIELE

1.	LPRINT ''AUSGABEPROGRAMM''	
2.	LPRINT ''HALLO'' ; ''TEST'' LPRINT CHR$(65) ; ''B'' , CHR$(67)	HALLOTEST AB C
4.	100 LPRINT ''A= '' ; A	A=0
5.	LPRINT ''B= '' ; B : LP. ''ENDE''	B=0 ENDE
6.	LP. : LP. : LP.	3 mal Zeilenvorschub (LF)

ERKLÄRUNG

NEW

Zum Löschen eines gespeicherten Programmes.

Der Befehl

1. löscht ein im Speicher stehendes Programm,
2. löscht alle Arten von Variablen,
3. kann programmiert werden.

BEISPIELE

1.
```
NEW
READY
```

2.
```
100   PRINT "A" : NEW
RUN
A
READY

LIST                                    Das Programm ist gelöscht

READY
```

ERKLÄRUNG

NOTE (NO.)

Zum Notieren der Sektor- und Bytenummern, von denen ab das nächste zu schreibende oder zu lesende Byte auf die Diskette geschrieben oder gelesen wird.

Der Befehl

1. setzt eine eröffnete Datei zur Diskette voraus,
2. enthält die Nummer des betreffenden Datenkanals hinter #,
3. stellt die Position fest, ab der innerhalb der Datei geschrieben oder gelesen wird,
4. liest die Nummer des Sektors dieser Position (1 bis 719) in die erste Variable,
5. liest die Nummer des Bytes dieser Position (0 bis 124) in die zweite Variable,
6. dient dem späteren wahlfreien Zugriff auf den Beginn von Datensätzen oder einzelnen Bytes in einer Datei, indem diese Stelle beim Schreiben oder Lesen notiert wird,
7. verwendet häufig eine zweite Datei auf derselben Diskette zum Ablegen der festgestellten Werte.

BEISPIELE

NOTE (NO.)

```
100  DIM A$(40)
200  OPEN #1 , 8 , 0 , "D : DATFIL.DAT"          Dateieröffnung
300  INPUT A$ : PRINT A$
400  IF LEN(A$) = 0 THEN 900                     letzte Eingabe: Taste  RETURN
500  NOTE #1 , X , Y                             notieren der Werte
600  PRINT #1 ; A$                               A$ wird mit EOL geschrieben
700  PRINT"SEKTOR # = " ; X                      Kontrollausdruck
750  PRINT"BYTE # = " ; Y                        Kontrollausdruck
800  GOTO 300
900  END                                         schließt die Datei

RUN                                              ? HIER IST OTTO IN FRANKFURT
                                                 SEKTOR # = 351
                                                 BYTE # = 0
                                                 ? TASTE RETURN
                                                 END
```

Die ausgegebenen Zahlen hängen jeweils von der Situation ab. Mehrere Eingaben machen und die ausgegebenen Zahlen aufschreiben. Die Datei wird für das Beispiel des Befehls **POINT** verwendet.

NUMERISCHE VARIABLEN

Variablen sind die Namen des Inhaltes von Speicherplätzen. Der Inhalt eines solchen Speicherplatzes kann mit Hilfe des Variablennamens gelesen, beschrieben oder verändert werden.

Variablennamen dürfen bis zu 120 Zeichen lang sein, müssen mit einem Buchstaben beginnen und dürfen nur Großbuchstaben und Ziffern enthalten. Es sollen keine **BASIC**-Befehlsworte als erster Teil des Namens verwendet werden.

Es können insgesamt bis zu 128 Variablennamen aller Variablenarten zugewiesen werden.

1. Eine numerische Variable enthält eine Zahl.

2. Die Zahl darf höchstens ±9.77009957 * 10E+99 gross sein.

3. Die Zahl muss mindestens ±1.0 * 10E−98 gross sein.

4. Der Variablenname kann beim Rechnen wie eine Zahl behandelt werden.

NUMERISCHE VARIABLEN

Zuweisung eines Wertes zu einer Variablen:

1. 100 A=0

2. 100 B=B+1 — Als Schrittzähler mit der Variablen B

3. 100 DRUCKEN=2000
   200 GOSUB DRUCKEN

4. 100 VERRECHNEN=1000
   200 GOTO VERRECHNEN + 10

5. 100 LET N=1700
   200 RESTORE(N−5)

6. ENDE=0
   ERROR −END[E] =0 — BASIC-Befehlswort END im Variablennamen.
   (**LET ENDE=0** behebt die Abweisung)

ON/GOSUB/RETURN (GOS./RET.)

Zur wahlweisen Fortsetzung des Programmes mit einem der in den hinter **GOSUB** stehenden Zeilennummern beginnenden **BASIC**-Unterprogramm, wenn die entsprechende Bedingung erfüllt ist ("bedingter Sprung").

Der Befehl enthält

1. zwischen **ON** und **GOSUB**
   a) die zu erfüllende Bedingung,
   b) den zu berechnenden Wert,
   c) die zu verwendende Variable.
   Der Ausdruck zwischen **ON** und **GOSUB** wird gerundet und muß zwischen 1 und 255 liegen. Wird hier z.B. die Zahl 2 gefunden, wird zu dem Unterprogramm gesprungen, dessen Zeilennummer an zweiter Stelle hinter **GOSUB** steht. Ist die Zahl gleich 0 oder größer als die Anzahl der aufgeführten Zeilennummer hinter **GOSUB**, wird nicht gesprungen und mit dem nächsten Befehl fortgefahren.

2. hinter **GOSUB** die Zeilennummern der möglichen Sprungziele durch Kommata getrennt.

Die Zeilennummern können auch als numerische Variablen angegeben werden. Nach der Rückkehr aus dem Unterprogramm wird das Programm mit dem nächsten Befehl fortgesetzt.

ON/GOSUB/RETURN (GOS./RET.)

1a.
```
100  ON A>B GOSUB 200
```

1b.
```
100  ON A+Z GOSUB 200 , 300 , 400
```

1c.
```
 50  ON A GOSUB 100 , 200 , 300
 60  ON A-3 GOSUB 400 , 500 , 600
```

2.
```
 100  A=1 : B=2 : EINS=1000 : ZWEI=2000
 200  PRINT "WIEVIEL " ; : INPUT K
 300  PRINT : TRAP 200
 400  ON K GOSUB EINS , ZWEI
 500  PRINT : GOTO 200
1000  PRINT A : RETURN
2000  PRINT B : RETURN
RUN
```

```
WIEVIEL ? Taste 2 [RETURN]
2
WIEVIEL ? Taste 1 [RETURN]
1
WIEVIEL ? . . . .
```

ERKLÄRUNG

ON/GOTO

Zur wahlweisen Fortsetzung des Programmes mit einer der hinter **GOTO** angegebenen **BASIC**-Zeilen, wenn die entsprechende Bedingung erfüllt ist ("bedingter Sprung").

Der Befehl enthält

1. zwischen **ON** und **GOTO**
   a) die zu erfüllende Bedingung,
   b) den zu berechnenden Wert,
   c) die zu verwendende numerische Variable.

2. hinter **GOTO** die Zeilennummern der möglichen Sprungziele (die Zeilennummern können auch als numerische Variablen angegeben werden). Durch Runden muß nun eine Zahl gefunden werden, die zwischen 1 und 255 liegt. Wird hier z.B. die Zahl 2 gefunden, wird zu der zweiten Zeilenangabe hinter **GOTO** gesprungen. Ist die Zahl gleich 0 oder größer als die Anzahl der aufgeführten Zeilennummern hinter **GOTO**, wird nicht gesprungen und mit der folgenden Zeile fortgefahren. Ist die Zahl <0 oder >255, führt dies zu Fehler 3 (siehe **ERROR**).

3. Sprung zu nicht vorhandenen Zeilennummern führt zu Fehler 12 (siehe **ERROR**) (siehe auch Befehle **ON GOSUB, GOSUB, GOTO**).

BEISPIELE

ON/GOTO

```
1a.   100  A=6 : B=5 : ON A>B GOTO 300
      200  STOP
      300  STOP
      RUN                                      STOPPED AT LINE 300
                                               (wahr=1 und unwahr=0, deshalb Sprung nach Zeile 300)

1b.   100  ON A+Z GOTO 200 , 300 , 400

1c.    50  ON A GOTO 100 , 200 , 300

2.     100  EINS=1000 : ZWEI=2000
       200  PRINT "WIEVIEL  " ; : INPUT K      WIEVIEL ? 1.5
       300  ON K GOTO EINS , ZWEI              HIER IST ZEILE 2000
       400  PRINT "K=0 ODER ZU GROSS."         READY
       500  PRINT : GOTO 200
      1000  PRINT "HIER IST ZEILE 1000" : END
      2000  PRINT "HIER IST ZEILE 2000" : END
      RUN                                      verschiedene Werte für K eingeben (auch negative)
```

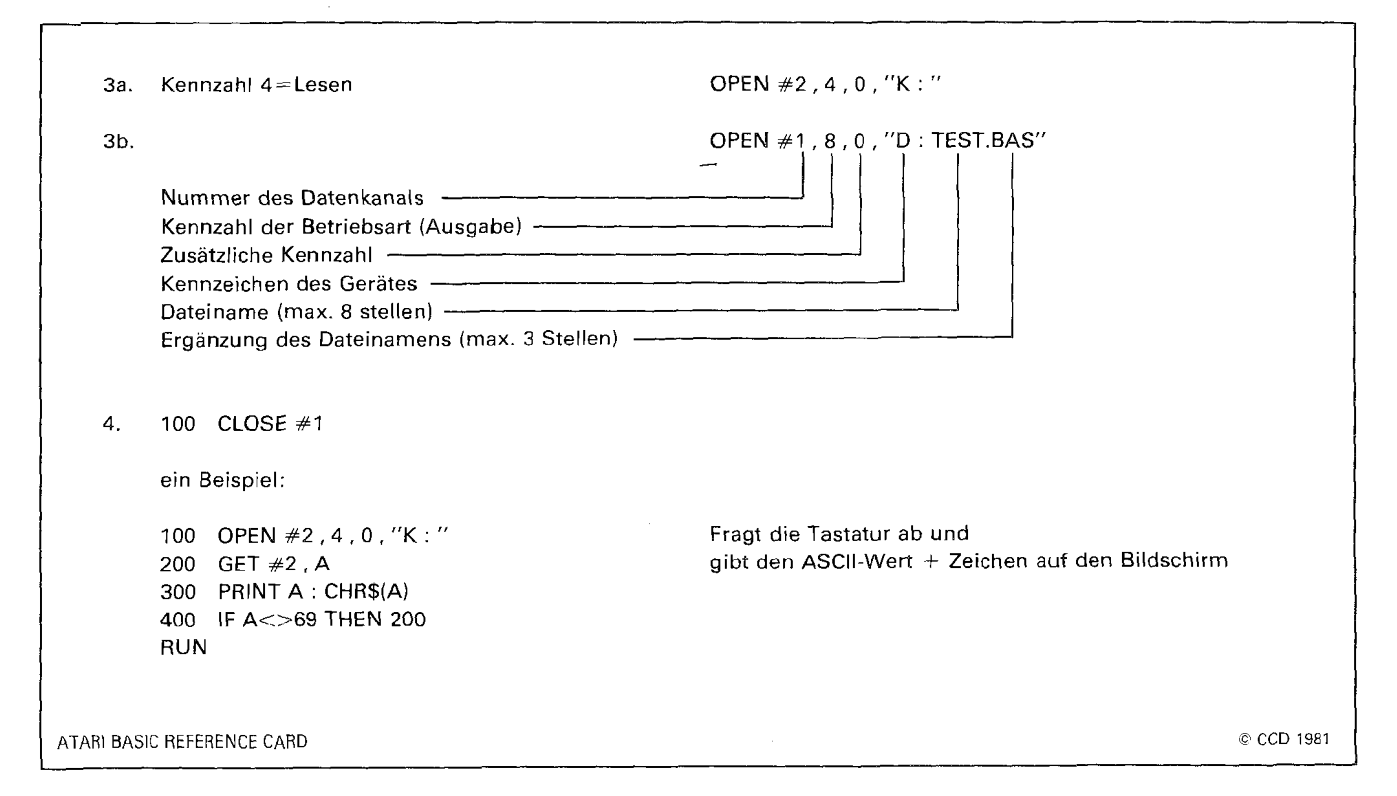
OPEN (O.)/CLOSE (CL.)

Zum Eröffnen und Schließen eines Datenkanals für die Ein- oder Ausgabe von Daten.

1. Es können Eingaben verarbeitet werden
   - von der Tastatur, Kennzeichen "K : ",
   - vom Editor (Betriebsart Programmeingabe), Kennzeichen "E : ",
   - vom Bildschirm, Kennzeichen "S : ",
   - vom Kassettenrekorder, Kennzeichen "C : ",
   - von der Diskettenstation, Kennzeichen "D : ",
   - vom Schnittstellenumsetzer ("RS-232-Interface"), Kennzeichen "R : "
2. Es können Daten ausgegeben werden an
   - den Editor (Textfenster), Kennzeichen "E : ",
   - den Bildschirm (Grafikfenster), Kennzeichen "S : ",
   - den Kassettenrekorder, Kennzeichen "C : ",
   - die Diskettenstation, Kennzeichen "D : ",
   - den Zeilendrucker, Kennzeichen "P : ",
   - den Schnittstellenumsetzer, Kennzeichen "R : ",
3. Ein Datenkanal wird eröffnet durch Angabe des Befehls **OPEN**, Nummer des Datenkanals, Kennzahl der Betriebsart, zusätzliche Kennzahl, Kennzeichen des angesprochenen Gerätes und eventuell Dateiname.
4. Ein eröffneter Datenkanal wird mit **CLOSE** # und der Nummer des Datenkanals geschlossen (abgeschaltet).

Die Kennzahl legt die Betriebsart eines Datenkanals fest: 4=Eingabe; 6=Disketteninhalt lesen; 8=Ausgabe; 12=Ein- und Ausgabe; 9=Ausgabe (anhängen an einen vorhandenen Datensatz).

OPEN (O.)/CLOSE (CL.)



3a. Kennzahl 4=Lesen — OPEN #2 , 4 , 0 , "K : "

3b. OPEN #1 , 8 , 0 , "D : TEST.BAS"

4. 100 CLOSE #1

ein Beispiel:

```
100  OPEN #2 , 4 , 0 , "K : "
200  GET #2 , A
300  PRINT A : CHR$(A)
400  IF A<>69 THEN 200
RUN
```

Fragt die Tastatur ab und
gibt den ASCII-Wert + Zeichen auf den Bildschirm

ERKLÄRUNG

Operatoren

Mathematische Operatoren sind Rechenanweisungen, die zwischen zwei Ausdrücken stehen und festlegen, wie diese verknüpft werden sollen.

1. Diese Ausdrücke können sein:
   - a. Zahlen oder
   - b. Variablen oder
   - c. zu berechnende Werte.
2. Arithmetische Operatoren sind die Zeichen
   - a. + für das Addieren,
   - b. − für das Subtrahieren oder das Vorzeichen einer Zahl
   - c. * für das Multiplizieren,
   - d. / für das Dividieren und
   - e. ∧ für das Exponenzieren.
3. Logische Operatoren ("wahr"=1 und "falsch"=0) sind
   - a. NOT für die logische Vertauschung von "wahr" und "falsch",
   - b. AND für das logische UND,
   - c. OR für das logische ODER,
   - d. < linker Ausdruck kleiner als der rechte,
   - e. > linker Ausdruck größer als der rechte,
   - f. = linker Ausdruck gleich groß wie der rechte,
   - g. <= linker Ausdruck kleiner oder gleich dem rechten,
   - h. >= linker Ausdruck größer oder gleich dem rechten,
   - i. <> linker Ausdruck nicht gleich dem rechten.

BEISPIELE

Operatoren

1a.	PRINT A+7	7
1b.	PRINT 3−1	2
1c.	PRINT 12*2	24
1d.	PRINT 12/2	6
1e.	PRINT 2∧2	3.99999996
2a.	A=0 : PRINT (NOT A)	1 (NOT immer in Klammern)
2b.	A=1 : B=1 : IF A AND B THEN PRINT"OK"	OK A und B sind wahr
2c.	A=1 : B=0 : IF A OR B THEN PRINT"OK"	OK A ist wahr
2d.	A=5 : B=6 : IF A<B THEN PRINT"OK"	OK weil 5 kleiner als 6
2e.	A=6 : B=5 : IF A>B THEN PRINT"OK"	OK weil 6 größer als 5
2f.	A=7 : IF A=7 THEN PRINT OK	OK weil A gleich 7
2g.	A=7 : IF A<=8 THEN PRINT"OK"	OK weil 7 nicht größer als 8
2h.	A=9 : IF A>=8 THEN PRINT"OK"	OK weil 9 nicht kleiner als 8
2i.	A=5 : B=7 : IF A<>B THEN PRINT"OK"	OK weil A nicht gleich B

Vorrang von Operatoren

Bei einem umfangreichen Rechenvorgang werden die innersten Klammerausdrücke zuerst berechnet; danach wird mit der nächsthöheren Klammerebene forgefahren.
Bei Berechnungen auf derselben Klammerebene werden die Operatoren in der folgenden Reihenfolge (Vorrangstufe) bearbeitet:

1. <, >, =, <=, >=, <> — bei der Verwendung mit String
2. − — negatives Vorzeichen
3. ^ — Exponentenbildung
4. * und / — Punktrechnung
5. + und − — Strichrechnung
6. <, >, =, <=, >=, <> — bei der Verwendung mit Zahlen, numerischen Variablen oder zu berechnenden Werten
7. NOT — logische Umkehrung, Negation
8. AND — logisches UND
9. OR — logisches ODER

Treten auf derselben Klammerebene mehrere Operatoren derselben Vorrangstufe auf, so werden diese in der Reihenfolge ihres Auftretens bearbeitet.

Vorrang von Operatoren

a.
```
PRINT 3*-4
```
−12
negatives Vorzeichen geht vor

b.
```
PRINT 3+4*5+6
```
29

c.
```
100  A=3 : B=3
200  IF A>2 AND B<4 THEN ?"OK"
```
OK
zuerst werden die beiden Bedingungen untersucht und dann geprüft, ob beide erfüllt sind.

d.
```
100  A=3 : B=1 : C=0
200  IF A AND B OR C THEN ?"OK"
```
A=3 ist <>0, deshalb "wahr"
OK
UND-Bedingung erfüllt

e.
```
100  A=0 : B=1 : C=1
200  IF A AND B OR C THEN ?"OK"
```
OK
UND-Bedingung nicht erfüllt, jedoch ODER-Bedingung

f.
```
100  A=0 : B=0 : C=1
200  IF A AND (B OR C) THEN °"OK"
READY
```
Klammer geht vor AND und ist erfüllt, UND-Bedingung nicht erfüllt, weil A=0

ERKLÄRUNG

PADDLE

Zum Feststellen der Stellung des Drehknopfes bei einem Drehregler.

Der Befehl

1. erlaubt die Auswahl eines der acht möglichen Drehregler, die jeweils zu zweit von links nach rechts an die vier Buchsen an der Vorderseite des Computers angeschlossen werden,

2. erwartet dazu
   a. eine Zahl
   b. eine Variable oder
   c. einen zu berechnenden Wert
   zwischen 0 und 7 (von links nach rechts) in der Klammer hinter **PADDLE,**

3. findet eine Zahl zwischen 1 (rechter Anschlag) und 228 (linker Anschlag), die die Stellung des Drehreglers angibt.

BEISPIELE

PADDLE

1.	100	PRINT PADDLE(0)	150 erster Drehregler
2a.	100	PRINT PADDLE(1)	100
2b.	100	A=1	
	200	PRINT PADDLE(A)	100
3.	100	PRINT PADDLE(2−1)	100

ERKLÄRUNG

PEEK

Zum Feststellen des Inhaltes einer Speicherstelle.

Die Funktion

1. liest den Inhalt einer Speicherzelle
   a. aus dem RAM-Bereich (Schreib-/Lesespeicher) oder
   b. aus dem ROM-Bereich (Nurlesespeicher),

2. gibt das Ergebnis als Dezimalzahl zwischen 0 und 255 aus,

3. verwendet zum Adressieren
   a. eine Zahl,
   b. eine Variable oder
   c. einen zu berechnenden Wert,
   die/der auf- bzw. abgerundet wird, in der Klammer hinter **PEEK,**

4. erwartet eine dezimale Adressenzahl zwischen 0 und 65535.

BEISPIELE

PEEK

1a.	PRINT PEEK(82)	2 (linker Rand)
1b.	PRINT PEEK(65000)	86 (im ROM-Betriebssystem)
3a.	PRINT PEEK(83)	39 (rechter Rand)
3b.	A=83 : PRINT PEEK(A)	39
3c.	PRINT PEEK(83.3)	39
4.	PRINT PEEK(70000)	ERROR 3 (Adresse zu groß)

ERKLÄRUNG

PLOT (PL.)

Zum Zeichnen eines Punktes.

Der Befehl

1. wirkt in allen Grafikmodi,
2. zeichnet eine Punkt an der hinter **PLOT** angegebenen Stelle im Grafikfenster,
3. enthält die horizontale Position an der ersten Stelle hinter **PLOT** [X] , Y,
4. enthält die vertikale Position an der zweiten Stelle hinter **PLOT** X , [Y],
5. zeichnet den Punkt in der Farbe und Helligkeit, die durch den vorangehenden **COLOR**-Befehl bestimmt worden ist,
6. ergibt in den Grafikmodi 3 – 11 nur dann einen sichtbaren Punkt, wenn **COLOR** ungleich 0 ist,
7. setzt den unsichtbaren Graphik-Cursor auf die angegebene Position,
8. verwendet als Nullpunkt für die Positionsangaben die linke, obere Bildschirmecke (X=0 , Y=0).

(siehe Befehle **DRAWTO** und **COLOR**)

BEISPIELE

PLOT (PL.)

100 GRAPHICS 3	Grafikmodus 3 bei den Farben werden die Anfangswerte des Computers verwendet
200 COLOR 1	Farbe 1 ist orange
300 PLOT 10 , 10	Punkt in orange
400 PLOT 15 , 15	Punkt in orange

ERKLÄRUNG

PLAYER MISSILE GRAFIK 1

Zur erleichterten Darstellung von bewegten Grafikfiguren ("Player und Missile" – "Spielerfigur und Geschoss") wird durch besondere Einrichtungen im ATARI 400/800 die PM-Grafik benutzt.
Die PM-Grafik erlaubt in jedem Grafikmodus die zusätzliche, bewegte Darstellung von vier Playern und vier, farblich entsprechenden Missiles bzw. von fünf Playern ohne Missiles.

Einschalten der PM-Grafik:	POKE 53277,3
Breite der PM-Objekte:	Zum Festlegen der Breite werden die Kennzahlen (0 und 2 = normale, 1 = doppelte und 3 = vierfache Breite) in die entsprechenden Adressen geschrieben.
Adressen Player 0–3:	53256 bis 53259
Adressen Missiles 0–3:	53260 bis 53263
Beispiel:	POKE 53257,1 (Player 1 in doppelter Breite)
Startadresse der PM-Daten:	Die Startadresse wird in der Adresse 54279 abgelegt. Sie wird als vielfaches von 256 (Anfang einer Speicher-"page") angegeben. Bei einzeiliger Auflösung muß sie gleichzeitig ein vielfaches von 2048 sein. Bei zweizeiliger Auflösung muß sie gleichzeitig ein vielfaches von 1024 sein.

PLAYER MISSILE GRAFIK 1

Vertikale Auflösung einzeilig:	POKE 559,46 Jedes Datenbyte ist einer Fernsehzeile zugeordnet. Das Objekt kann aus bis zu 256 Bytes bestehn.
Vertikale Auflösung zweizeilig:	POKE 559,62 Jedes Datenbyte ist zwei Fernsehzeilen zugeordnet. Das Objekt kann aus bis zu 128 Bytes bestehn.
Horizontale Position:	Die horizontale Position des Objektes wird durch die Kennzahl (0=links und 256=rechts) bestimmt. Diese wird in die entsprechende Adresse geschrieben.
Adressen Player 0–3:	53248 bis 53251
Adressen Missiles 0–3:	53252 bis 53255
Farbeimer für die PM-Objekte Adressen für Player und Missiles 0–3:	704 bis 707
Vorrang der Darstellungen:	Unabhängig von der normalen Grafik kann durch schreiben in die Adresse 623 bestimmt werden, welche Darstellung vor welcher erscheint.

POKE 623,1 : alle Player vor der normalen Grafik
POKE 623,4 : normale Grafik vor allen Playern
POKE 623,2 : Player 0 und 1 vor der normalen Grafik, Player 2 und 3 hinter beiden.
POKE 623,8 : 2 Farbregister (708 und 709) vor allen Playern, die beiden anderen Farbregister (710 und 711) hinter beiden.

PLAYER MISSILE GRAFIK 2

Speicherorganisation der PM-Daten:

Die Anzahl der im Speicher vorhandenen Pages (Seiten zu je 256 Bytes) findet man mit PEEK(106). Die PM-Grafikdaten werden 2048 Bytes tiefer begonnen. Die Seitenzahl der Startadresse wird in Adresse 54279 geschrieben.

einzeilig	zweizeilig	von – bis
	Player 3	1792 – 2048
	Player 2	1536 – 1791
	Player 1	1280 – 1535
	Player 0	1024 – 1279
Player 3	M3 M2 M1 M0	896 – 1023
Player 2	M3 M2 M1 M0	768 – 895
Player 1		640 – 767
Player 0		512 – 639
M3 M2 M1 M0		384 – 511
ungenutzt		0 – 383

POKE 53279,PEEK(106) – 8 reserviert den Speicherbereich und PMSTART = 256*PEEK(106) – 8 ist die Listenbasis, von der aus die Speicherbereiche beschrieben werden.

PLAYER MISSILE GRAFIK 2

Aufbereitung der PM-Daten:

Jeder Player wird im Speicher als Datenwort von 8 Bits dargestellt. Jede Missile wird als Doppelbit dargestellt. Durch ihre Anordnung im Speicher können die vier Missiles auch wahlweise als 5. Player verwendet werden.

	Bit	7	6	5	4	3	2	1	0
1 Byte:									
	Wertigkeiten	128	64	32	16	8	4	2	1

Eine 1 im jeweiligen Bit wird mit der für den Player gewählten Farbe und Helligkeit dargestellt, eine 0 in der Farbe des Hintergrunds. Die Wertigkeiten für die Bits, die in der Playerfarbe dargestellt sein sollen, werden zu einer Dezimalzahl addiert:

	grafische Darstellung	Bitmuster	Dezimalzahl
Byte 1		00011000	16+8=24
Byte 2		00111100	32+16+8+4=60
Byte 3		00100100	32+4=36

PLAYER MISSILE GRAFIK 3

Mit diesem Programmbeispiel wird ein Flugzeug als Player definiert, das mit einem Steuerknüppel an der linken Buchse geführt wird.

Programm	Erläuterung
100 SETCOLOR 2 , 0 , 0	Hintergrund schwarz
200 X=130	horizontale Ausgangsposition für das Flugzeug. Zwischen 48 für links und 207 für rechts
300 Y=70	vertikale Ausgangsposition für das Flugzeug, zwischen 4 für oben und 123 für unten wenn zweizeilig aufgelöst und 8 für oben und 247 für unten wenn einzeilig aufgelöst wird.
400 A=PEEK(106)−8 : POKE 54279 , A : PMSTART=A*256	reserviert Speicherplatz für die PM-Daten
500 POKE 559 , 46	zweizeilige Auflösung
600 POKE 53277 , 3	PM-Grafik einschalten
700 POKE 53248 , X	Player 0 auf Position X setzen
800 FOR J=PMSTART+512 TO PMSTART+640 : POKE J , 0 : NEXT J	Speicherbereich löschen in den PM-Grafik soll
900 POKE 704 , 88	Farbe lila für Player 0
1000 FOR J=PMSTART+512+Y TO PMSTART +519+Y : READ A : POKE J , A : NEXT J	PM-Daten in den Speicher schreiben
1100 DATA 8 , 24 , 49 , 51 , 255 , 48 , 24 , 8	PM-Daten des Flugzeugs

PLAYER MISSILE GRAFIK 3

```
1200  A=STICK(0)
1300  IF A=15 THEN 1200
1400  IF A=11 THEN X=X-1 : POKE 53248 , X
1500  IF A=7 THEN X=X+1 : POKE 53248 , X
1600  IF A=13 THEN FOR J=10 TO 0 STEP-1 :
      POKE PMSTART+512+Y+J , PEEK(PMSTART
      +511+Y+J) : NEXT J : Y=Y+1
1700  IF A=14 THEN FOR J=0 TO 10 :
      POKE PMSTART+511+Y+J , PEEK(PMSTART
      +512+Y+J) : NEXT J : Y=Y-1
1800  GOTO 1200
```

Steuerknüppel 0 lesen und
warten bis er bewegt wird:
wenn nach links, dann Flugzeug nach links bewegen,
wenn nach rechts, dann Flugzeug nach rechts bewegen,
wenn nach oben, dann Flugzeug nach oben bewegen

wenn nach unten dann Flugzeug nach unten bewegen

Schleife zum Abfragen des Steuerknüppels.

ERKLÄRUNG

POINT (P.)

Zum Festlegen der Sektor- und Bytenummern, von denen ab das nächste zu schreibende oder zu lesende Byte auf die Diskette geschrieben oder gelesen wird.

Der Befehl

1. setzt eine zu eröffnende Datei zur Diskette voraus,
2. enthält die Nummer des betreffenden Datenkanals hinter #,
3. legt die Position fest, ab der innerhalb der Datei geschrieben oder gelesen werden soll,
4. setzt den Sektor-Zeiger auf den Wert der ersten Variablen (1 bis 719),
5. setzt den Byte-Zeiger auf den Wert der zweiten Variablen (1 bis 124),
6. dient dem wahlfreien Zugriff auf den Beginn von Datensätzen oder einzelnen Bytes in einer Datei, indem der Ort des Beginns des Schreib- oder Lesevorganges festgelegt werden kann,
7. findet häufig eine Datei auf derselben Diskette vor, in der die notierten Startwerte der Datensätze abgelegt sind.

POINT (P.)

```
100  DIM A$(40)
200  OPEN #1 , 4 , 0 , "D : DATFIL.DAT"        Dateieröffnung
300  PRINT"SEKTOR # = " ; : INPUT X           Eingabe Sektor-Nr.
400  PRINT"BYTE # = " ; : INPUT Y             Eingabe Byte-Nr.
500  POINT # , X , Y                          Zeiger auf gewünschte Lesestelle setzen
600  INPUT #1 ; A$ : PRINT A$                 liest A$ von Startpunkt bis EOL in A$
700  PRINT : GOTO 300
```

Dieses Beispiel setzt voraus, daß die Datei aus dem Beispiel des Befehls **NOTE** auf der Diskette vorhanden ist. Die dort aufgeschriebenen Werte hier verwenden.

```
RUN                                           SEKTOR # ? 351
                                              BYTE # ? 0
                                              HIER IST OTTO IN FRANKFURT
```

POKE

Zum Schreiben in eine Speicherzelle.

Der Befehl

1. schreibt in die Speicherzelle, deren Adresse als
   a. Zahl,
   b. Variable oder
   c. zu berechnenden Wert
   hinter **POKE** steht,

2. schreibt
   a. die Zahl,
   b. die Variable oder
   den zu berechnenden Wert,
   in die Speicherzelle, die (durch Komma getrennt) hinter ihrer Adresszahl steht,

3. erwartet eine dezimale Adresszahl zwischen 0 und 65535,

4. erwartet einen dezimalen Wert zwischen 0 und 255, um ihn in die angegebene Zelle zu schreiben,

5. verändert nur den Inhalt von Speicherzellen im RAM-Bereich.

BEISPIELE

POKE

1a.	POKE 82 , 0	(linken Rand auf 0)
1b.	A=82 : POKE A , 0	
1c.	POKE 82 , (10−10)	
2a.	POKE 83 , 30	(rechten Rand auf 30)
2b.	B=30 : POKE 83 , B	
2c.	POKE 83 , (40−10)	
3.	POKE 70000 , 10	ERROR 3 Adresszahl zu groß
4.	POKE 82 , 300	ERROR 3 zu schreibender Wert zu groß
5.	100 PRINT PEEK(65000) 200 POKE 65000 , 100 300 PRINT PEEK (65000)	86 Speicherzelle nicht RAM 86 (konnte nicht verändert werden da ROM-Bereich)

ERKLÄRUNG

POP

Zur Rückkehr aus einem **BASIC**-Unterprogramm zu dem aufrufenden Programm, ohne den Befehl **RETURN** auszuführen.

Wenn aus einem Unterprogramm mit **GOSUB** zu einem **BASIC**-Unterprogramm gesprungen wird, wird in dem sogenannten "Stack-Speicher" die Zeilennummer gespeichert, an der das Hauptprogramm verlassen wurde. Am Ende des Unterprogramms bewirkt der Befehl **RETURN** die Rückkehr zum Hauptprogramm und dessen Fortsetzung bei der gespeicherten Zeilennummer (Normalfall).
Soll nun von einer beliebigen Stelle des Unterprogrammes zum Hauptprogramm zurückgesprungen werden (jedoch nicht zu der Zeile, an der das Hauptprogramm verlassen wurde), muß der Sprung mit den Befehlen **POP** und **GOTO** ausgeführt werden.

Die Befehle **POP** und **RETURN** entfernen jeweils die gespeicherte Zeilennummer aus dem Stack-Speicher.

1. Für jedes **GOSUB**, das nicht über **RETURN** zum Hauptprogramm zurückführt, muß ein **POP**-Befehl gegeben werden.

2. Der **POP**-Befehl muß sich im Zuge des weiteren Programmablaufes hinter **GOSUB** befinden, um wirksam zu werden.

POSITION (POS.)

Zum Setzen des unsichtbaren Grafik-Cursors oder des Text-Cursors auf eine bestimmte Stelle des Bildschirms.

Der Befehl

1. wirkt in allen Grafikmodi (0-11) und Textmode,
2. stellt den unsichtbaren Grafik-Cursor oder den Text-Cursor an die hinter **POSITION** angegebene Stelle auf den Bildschirm,
3. enthält die horizontale Position an der ersten Stelle hinter **POSITION** [X] , Y,
4. enthält die vertikale Position an der zweiten Stelle hinter **POSITION** X , [Y],
5. wird in der Regel von dem Befehl **PRINT** gefolgt,
6. bewegt den Cursor erst, wenn ein Ein- oder Ausgabebefehl, der den Bildschirm betrifft, ausgeführt wird,
7. verwendet als Nullpunkt für die Positionsangaben die linke, obere Bildschirmecke (X=0 , Y=0).

PRINT (PR oder ?)

Zur Ausgabe von Informationen auf den Bildschirm oder zu anderen Ausgabegeräten (siehe Befehl **LPRINT**).

Es können ausgegeben werden:

1. Texte aus Buchstaben und Zeichen im Klartext (z.B. "Otto"),
2. Texte aus Buchstaben und Zeichen als Stringvariablen (z.B. A$),
3. Zahlenwerte aus Variablen (z.B. A),
4. Zahlenwerte als Ergebnis eines Rechenvorganges (z.B. A+B),
5. Leerzeilen (nur PRINT angeben),
6. Ausgaben an andere Ausgabegeräte als den Bildschirm (siehe **OPEN/CLOSE**),
7. Cursorbewegungen,
8. Bildschirmlöschung,
9. Sonderzeichen (Summer, EOF etc., siehe ATASCII-Tabelle).

Jedes PRINT leitet eine neue Zeile ein. Soll in derselben Zeile weitergemacht werden, wird mit

10. ; an der folgenden Stelle derselben Zeile weitergedruckt,
11. , an der nächsten Tabulator-Position (siehe Taste [TAB]) weitergedruckt.

Pseudo-Grafik

Der Teil des Zeichensatzes, dessen ATASCII-Codezahlen zwischen 0 und 31 ("Steuerzeichen") liegen, bzw. 96 oder 123 sind. Er enthält Zeichen wie z.B. Balken verschiedener Dicke für waagerechte und senkrechte Verwendung, die Spielkartensymbole Kreuz, Pik, Herz und Karo usw.

Die Zeichen können

1. über den Befehl **PRINT** ausgegeben werden,

2. über Buchstaben- bzw. Punkt-, Komma-, oder Semikolon.Tasten eingegeben werden, wenn gleichzeitig die Taste [CTRL] gedrückt wird (siehe auch hinterer Deckel des Handbuches),

Die Tasten können

3. auf Dauer umgeschaltet werden, wenn gleichzeitig die Tasten [CTRL] und [CAPS/LOWR] gedrückt werden,

4. wieder auf kleine Buchstaben zurückgeschaltet werden, wenn die Taste [CAPS/LOWR] gedrückt wird,

5. wieder auf große Buchstaben zurückgeschaltet werden, wenn gleichzeitig die Tasten [SHIFT] und [CAPS/LOWR] gedrückt werden,

6. verwendet wird die Pseudografik u.a. für die einfache Herstellung von Textumrandungen.

ERKLÄRUNG

PTRIG

Zum Feststellen, ob der Druckknopf eines Drehreglers gedrückt ist.

Der Befehl

1. erlaubt die Auswahl eines der acht möglichen Drehregler, die jeweils zu zweit (von links nach rechts) an die vier Buchsen an der Vorderseite des Computers angeschlossen werden,

2. erwartet dazu
   a. eine Zahl,
   b. eine Variable oder
   c. einen zu berechnenden Wert
   zwischen 0 und 7 (von links nach rechts) in der Klammer hinter **PTRIG**,

3a. findet die Zahl 0, wenn der Knopf gedrückt ist,

3b. findet die Zahl 1, wenn der Knopf nicht gedrückt ist.

ERKLÄRUNG

PUT (PU.)

Zur Ausgabe eines Bytes über einen eröffneten Datenkanal.

Der Befehl

1. setzt einen mit **OPEN** eröffneten Datenkanal voraus (Kennzahl 8, 12 oder 9),

2. muß mit # und der Nummer des Datenkanals versehen werden, über den ausgegeben werden soll,

3. gibt ein Byte aus (8-Bit-Parallelwort, entspricht einer Zahl zwischen 0 und 255), das
   a) eine Zahl,
   b) eine Variable
   c) oder ein zu berechnender Wert sein kann.

(siehe Befehl **GET**)

ERKLÄRUNG

RAD

Zum Umschalten der Winkelfunktionen auf das Bogenmaß.

Der Befehl

1. schaltet die Winkelfunktionen auf das Bogenmaß (Kreisumfang entspricht zweimal π),
2. wird nicht aufgehoben, wenn die Taste SYSTEM RESET gedrückt wird,
3. ist eingeschaltet, wenn der Coputer eingeschaltet wird.

READ (REA.)/DATA (D.)

Zur Zuweisung von Werten, die aus einer **DATA**-Liste gelesen werden, zu Variablen.

Der Befehl

1. liest nacheinander die durch Kommas getrennten Werte oder Zeichen aus den **DATA**-Zeilen in eine oder mehrere, durch Kommas getrennte Variablen, die hinter **READ** angegeben werden,

2. stellt bei jedem Lesen eines Wertes oder Zeichens den **READ/DATA**-Zeiger um eins weiter (siehe Befehl **RESTORE**),

3. erwartet, daß nicht mehr Werte durch **READ** gelesen werden sollen, als in der **DATA**-Liste noch vorhanden sind (Listenende markieren),

4. erwartet, daß die Variablenart hinter **READ** den zu lesenden Werten entspricht,

5. wird in der Regel programmiert (dabei kann die **DATA**-Liste irgendwo im Programm stehen),

6. kann auch direkt eingegeben werden (dabei muß eine zu lesende **DATA**-Zeile bereits vorher im Programm ausgeführt worden sein),

7. läßt hinter **DATA** als zu lesende Daten keine Kommas zu; Anführungszeichen werden ausgegeben.

ERKLÄRUNG

REM (R. oder .LEERTASTE)

Zur Darstellung von Erläuterungen ("REMARKS") im Programm.

Der Befehl

1. erlaubt die Darstellung von Erläuterungen in normalen **BASIC**-Zeilen,
2. verhindert die Bearbeitung des Teiles der **BASIC**-Zeile, der hinter **REM** steht,
3. kann auch dazu verwendet werden, beim Austesten eine **BASIC**-Zeile durch davorsetzen von **REM** zu sperren, ohne sie zu löschen,
4. wird jedoch beim Auslisten mit ausgegeben,
5. kann nur programmiert werden.

RESTORE (RES.)

Zum Setzen des READ/DATA-Zeigers auf die DATA-Zeile, mit der der Lesevorgang beginnen soll (siehe Befehl **READ/DATA**).

Der Befehl

1. setzt den READ/DATA-Zeiger auf den ersten Wert in der ersten DATA-Zeile des Programms, wenn nur **RESTORE** allein angegeben wird,

2. setzt den READ/DATA-Zeiger auf den ersten Wert der DATA-Zeile, deren Zeilennummer hinter **RESTORE** angegeben wird,

3. erlaubt den wiederholten Gebrauch derselben DATA-Liste,

4. erlaubt das gezielte, schnelle Auffinden von Daten in einer DATA-Liste, wenn die Zeilennummer bekannt ist, in der der Wert steht.

ERKLÄRUNG

Taste RETURN

Zum Beenden einer Zeile bei der Eingabe.

Die Taste

1. beendet die Eingabe einer BASIC-Zeile bei der Programmeingabe; ihre Funktion ist hier mit **POKE** 764,12 programmierbar; danach erscheint **"READY"**, die Ausführung erscheint eine Zeile unter der Meldung **READY**,

2. beendet eine Zeile bei der Direkteingabe; danach erscheint **"READY"** nicht,

3. beendet eine Eingabe während der Programmdurchführung als Antwort auf den Befehl **INPUT**; danach wird mit der Programmdurchführung fortgefahren,

4. hat im ATASCII-Code die Bezeichnung **EOL** ("**END-OF-LINE**") und die Codezahl 155,

5. bringt den Cursor nach Abschluß der Zeile zum Beginn der folgenden Zeile.

   (siehe Betriebsarten und BASIC Zeilen-Format)

ERKLÄRUNG

RND

Zum Erzeugen einer Zufallszahl.

Die Funktion

1. erzeugt eine Zufallszahl zwischen 0 und 1,
2. erzeugt nie eine 1,
3. benötigt in der Klammer hinter **RND** immer eine Dummy-Variable oder -Zahl, die aber ohne Einfluß bleibt,
4. wird häufig in Spielprogrammen in Verbindung mit anderen **BASIC**-Statements als Würfel verwendet.

RUN (RU.)

Der Befehl

1. beinhaltet **CLR** und **RESTORE**,
2. setzt den Stack zurück (siehe Befehle **FOR/NEXT** und **GOSUB**),
3. schliesst alle offenen Datenkanäle und schaltet alle Tongeneratoren aus,
4. beginnt die Programmdurchführung eines gespeicherten Programmes, wenn nach **RUN** keine Dateiangaben gemacht worden sind oder
5. beginnt die Programmdurchführung, nachdem das Programm von dem hinter **RUN** in den Dateiangaben bezeichneten Datei geladen worden ist, wenn das betreffende Programm mit **SAVE** gespeichert worden war,
6. kann das Programm nur bei der niedrigsten Zeilennummer starten (siehe Befehl **GOTO**),
7. kann programmiert werden.

Die Programmdurchführung läuft bis zu einem **STOP**- oder **END**-Befehl, bis zu einem Fehler (siehe Befehl **TRAP**) oder bis die Tasten [BREAK] oder [SYSTEM RESET] gedrückt wurden.

ERKLÄRUNG

SAVE (S.)

Zum Schreiben eines im Speicher stehenden Programmes in übersetzter Form zum Kassettenrekorder oder zur Diskettenstation.

Der Befehl

1. verändert ein im Speicher stehendes Programm samt Variablen nicht,
2. schreibt das Programm in übersetzter Form (Tokenformat) (siehe Befehle **CSAVE** und **LIST**),
3. gibt an die Kassette aus, wenn "C : " hinter **SAVE** angegeben wurde,
4. gibt an die Diskette aus, wenn "D : " hinter **SAVE** angegeben wurde,
5. muß für die Diskette mit einem Programmnamen versehen werden,
6. wird in der Regel direkt eingegeben,
7. kann auch programmiert werden,
8. schreibt langsamer als **CSAVE**, weil die Pausen zwischen den Daten-Blöcken länger sind.

ERKLÄRUNG

SETCOLOR (SE.)

Zum Festlegen der Farbe und Helligkeit, die in einem bestimmten Farbregister ("Farbeimer") gespeichert werden soll.

Der Befehl

1. füllt die Farbregister ("Farbeimer") mit den gewünschten Farb- und Helligkeitswerten (bereits dargestellte farbige Bildteile verändern ihre Farbe durch nachträgliches Ändern der Werte im Farbregister),
2. wirkt in allen Grafikmodi,
3. enthält die Kennzahlen für
   a. das Farbregister (0 bis 4) an erster Stelle
   b. die Farbe (0 bis 15) an zweiter Stelle
   c. die Helligkeit (gerade Zahl von 0=dunkel bis 14=hell) an dritter Stelle
   hinter **SETCOLOR** a, b, c,
4. die folgenden Anfangswerte werden vom Computer in die Farbregister geschrieben

a.	SETCOLOR 0 , 2 , 8	(orange),
b.	SETCOLOR 1 , 12 , 10	(hellgrün),
c.	SETCOLOR 2 , 9 , 4	(dunkelblau), Hintergrundfarbe im Grafikmode 0 und für alle Textfenster
d.	SETCOLOR 3 , 4 , 6	(rot),
e.	SETCOLOR 4 , 0 , 0	(schwarz). Randfarbe im Grafikmode 0 bzw. Hintergrundfarbe des Grafikfensters

(Diese Farben werden automatisch nach dem Einschalten des Computers und nach Betätigung der Taste [SYSTEM RESET] gesetzt. Siehe auch Befehle **GRAPHICS 0-11** und **COLOR**.)

ERKLÄRUNG

SGN

Zum Feststellen des Vorzeichens einer Zahl.

Die Funktion

1. stellt das Vorzeichen fest von
   a. einer Zahl
   b. einer Variablen oder
   c. einem zu berechnenden Wert,

2. ergibt −1 , wenn der Wert in Klammern kleiner als 0 ist,

3. ergibt 0 , wenn der Wert in Klammern gleich 0 ist,

4. ergibt +1 , wenn der Wert in Klammern größer als 0 ist.

ERKLÄRUNG

Taste SHIFT

Zum Umschalten der Funktion einiger Tasten (erste Umschaltebene).

Wird die Taste [SHIFT] gleichzeitig gedrückt mit

1. einer Buchstabentaste, wenn auf Kleinbuchstaben geschaltet war, so wird der entsprechende Großbuchstabe dargestellt,
2. der Taste [TAB], so wird an die Cursorfunktion eine Tabulatormarke gesetzt,
3. der Taste [CLEAR], so wird der Bildschirm gelöscht,
4. der Taste [INSERT], so wird an der Cursorposition eine Leerzeile eingeschoben; der restliche Bildschirminhalt wird nach unten verschoben; dabei geht die Zeile am unteren Rand verloren,
5. der Taste [DELETE BACK'S], so wird die Zeile an der Cursorposition entfernt; der Rest des Bildschirminhalts rückt eine Zeile nach oben.

ERKLÄRUNG

SIN

Zum Errechnen des Sinus.

Die Funktion

1. errechnet den Sinus
   a. der Zahl,
   der Variablen oder
   c. des zu berechnenden Wertes,

die,der in der Klammer hinter **SIN** steht,

2. ergibt immer Werte zwischen −1 und +1,

3. kann den Winkel im Bogenmaß oder in Grad verarbeiten.

(siehe Befehle **DEG** und **RAD**)

ERKLÄRUNG

SOUND (SO.)

Zum Erzeugen von Tönen, Klängen und Geräuschen.

Der Befehl

1. erlaubt die Anwahl eines der vier vorhandenen Tongeneratoren durch die erste Zahl (0 bis 3) hinter **SOUND,**
2. erlaubt die Steuerung der Tonhöhe durch die zweite Zahl (0≙hoch bis 255≙tief) hinter **SOUND,**
3. erlaubt die Steuerung des Verzerrungsgrades des Tones durch die dritte Zahl (0 bis 14) hinter **SOUND,**
4. erlaubt die Steuerung der Lautstärke durch die vierte Zahl (1≙leise bis 15≙laut) hinter **SOUND,**
5. gestattet das Ausschalten des angewählten Tongenerators durch einen neuen **SOUND**-Befehl mit der Lautstärke 0; sollen alle Tongeneratoren gleichzeitig ausgeschaltet werden, wird der Befehl **END** benutzt,
6. erlaubt die gleichzeitige Benutzung aller vier Tongeneratoren.

Es können die genannten Zahlen an allen vier Stellen durch Variablen oder zu berechnende Werte ersetzt werden. Die Töne können über den Lautsprecher des FS-Gerätes gehört, und beim ATARI 800 zusätzlich an der Monitorbuchse abgenommen werden.

ERKLÄRUNG

SOUND (SO.), Erg. 1

Mit den vier Tongeneratoren können musikalische Noten gespielt werden, wenn für die zweite Zahl hinter **SOUND** die entsprechenden Werte eingesetzt werden. **POKE 53768,1** bzw. **POKE 53768,4** verschieben den Tonbereich nach unten bzw. oben.

Tonhöhe	Zahl	Tohnhöhe	Zahl	Tonhöhe	Zahl
tiefes C	243	kleines C	121	hohes c′	60
Cis	230	cis	114	cis′	57
D	217	d	108	d′	53
Dis	204	dis	102	dis′	50
E	193	e	96	e′	47
F	182	f	91	f′	45
Fis	173	fis	85	fis′	42
G	162	g	81	g′	40
Gis	153	gis	76	gis′	37
A	144	a	72	a′	35
Ais	136	ais	68	ais′	33
H	128	h	64	h′	31
C	121	c′	60	c″	29

ERKLÄRUNG

SQR

Zum Errechnen der Quadratwurzel einer Zahl.

Die Funktion

1. errechnet die Quadratwurzel der Zahl, die in der Klammer hinter **SQR** steht,

2. setzt voraus, daß der Wert in Klammern positiv ist.

Speicherbereich im Computer, in dem von der CPU u. a. beim Einsprung in Unterprogramme die Rücksprungadressen abgelegt werden.

Der Stack

1. liegt beim ATARI 400/800 im Speicherbereich zwischen 100 und 1FF (Hexadezimal) bzw. 256 und 511 (dezimal),
2. wird von der CPU mit Hilfe eines eigenen Zählers (des Stackpointers) beschrieben und gelesen,
3. wird von der höchsten Speicherstelle nach tieferen fortschreitend beschrieben und gelesen,
4. arbeitet nach dem Prinzip "als Erster hinein, als Letzter hinaus",
5. übernimmt beim Einsprung in ein Unterprogramm die Rücksprungadresse, damit nach abgearbeitetem Unterprogramm das Hauptprogramm an der richtigen Stelle fortgesetzt werden kann,
6. enthält noch die alte Rücksprungadresse, wenn ein Unterprogramm nicht mit dem vorgesehenem Befehl (**RTS** bzw. **RETURN**) verlassen wurde, was bei geschachtelten Unterprogrammen beim nächsten **RTS-** bzw. **RETURN**-Befehl mit einem Rücksprung zur falschen Stelle endet (siehe Befehl **POP**),
7. kann zur Übergabe von Parametern zwischen Unterprogrammen benutzt werden (siehe Befehl **USR**).

ERKLÄRUNG

STATUS (ST.)

Zum Feststellen des Betriebszustandes eines Datenkanals (z.B. mit dem Kennzeichen "D :", "P :" oder "R :").

Der Befehl

1. setzt einen eröffneten Datenkanal voraus,
2. verwendet die Nummer des Datenkanals hinter #,
3. liest die Kennzahl des in dem genannten Datenkanal aufgetretenen Fehlers in die Variable,
4. erkennt Fehler im Zusammenhang mit
   a. der Diskettenstation,
   b. dem Drucker,
   c. dem Schnittstellenumsetzer.

(siehe **ERROR**)

Zum Feststellen der Stellung eines Steuerknüppels.

Der Befehl

1. erlaubt die Auswahl eines der vier möglichen Steuerknüppel, die von links nach rechts an die vier Buchsen an der Vorderseite des Computers angeschlossen werden,

2. erwartet dazu
   a. eine Zahl,
   b. eine Variable oder
   c. einen zu berechnenden Wert
   zwischen 0 und 3 (von links nach rechts) in der Klammer hinter **STICK,**

3. findet die angegebene Zahl, wenn der Steuerknüppel in die angegebene Richtung gedrückt wird:

15	in Ruhe
14	nach vorn
13	zurück
7	nach rechts
11	nach links
6	rechts vorn
10	links vorn
9	links zurück
5	rechts zurück

ERKLÄRUNG

STOP (STO.)

Zur Unterbrechung der Programmausführung an dieser Stelle.

Der Befehl

1. unterbricht die Programmausführung an dieser Stelle,
2. löscht das Programm und die Variablen nicht,
3. läßt Datenkanäle, Tongeneratoren und Grafikmodus unverändert,
4. wird im Programm beim Austesten verwendet,
5. kann nur programmiert werden (siehe Taste [BREAK]),
6. kann wieder rückgängig gemacht werden (siehe Befehl **CONT**),
7. gibt nach der Programmunterbrechung die Zeile an, in der das Programm angehalten hat,
8. gibt die Tastatur wieder für den Benutzer zur Programmierung frei.

ERKLÄRUNG

STR$

Zum Umwandeln einer Zahl in eine String-Variable.

Die Funktion

1. macht eine String-Variable aus der Zahl, die in der Klammer hinter **STR$** steht,
2. kann wie eine String-Variable behandelt werden,
3. darf in einer Vergleichsoperation nur einmal vorkommen.

(siehe Befehl **VAL**)

ERKLÄRUNG

STRIG

Zum Feststellen, ob der Feuerknopf eines Steuerknüppels gedrückt ist.

Der Befehl

1. erlaubt die Auswahl eines der vier möglichen Steuerknüppel, die von links nach rechts an die vier Buchsen an der Vorderseite des Computers angeschlossen werden,

2. erwartet dazu
   a. eine Zahl,
   b. eine Variable oder
   c. einen zu berechnenden Wert
   zwischen 0 und 3 (von links nach rechts) in der Klammer hinter **STRIG,**

3. findet die Zahl
   a. 0, wenn der Feuerknopf gedrückt ist,
   b. 1, wenn der Feuerknopf nicht gedrückt ist.

STRING-VARIABLEN

Variablen sind die Namen des Inhaltes von Speicherplätzen. Der Inhalt eines solchen Speicherplatzes kann mit Hilfe des Variablennamens gelesen, beschrieben oder verändert werden.

Variablennamen dürfen bis zu 120 Zeichen lang sein, müssen mit einem Buchstaben beginnen und dürfen nur Grossbuchstaben und Ziffern enthalten. Es sollen keine **BASIC**-Befehlsworte als erster Teil des Namens verwendet werden.

Es können insgesamt bis zu 128 Variablennamen aller Variablenarten zugewiesen werden.

1. Eine STRING-VARIABLE enthält Text (z.B. "Otto"), also Buchstaben, Ziffern oder Sonderzeichen ("STRING"). Ihr Name endet immer mit $, sie muss "dimensioniert" werden (siehe Befehl **DIM**).

2. In einer **STRING-VARIABLEN** können nur soviele Zeichen abgelegt werden, wie mit dem **DIM**-Befehl vorgegeben wurde.

3. Der Variablenname kann bei der Ausgabe mit einem **PRINT**-Befehl wie ein Text behandelt werden.

4. Es können Untermengen angesprochen und String-Verknüpfungen durchgeführt werden

Strings teilen

Zum Herstellen einer Teilstringvariablen aus einer Stringvariablen.

Der Befehl

1. erzeugt eine Teilstringvariable aus der angegebenen Stringvariablen,

2. beginnt den Teilstring an der Stelle des Strings, die durch die Zahl in Klammern hinter dem Stringnahmen angegeben wird,

3. beendet den Teilstring
   a. an der Stelle des Strings, die durch die zweite Zahl in der Klammer hinter dem Stringnahmen angegeben wurde oder
   b. am Ende des Strings, wenn in der Klammer eine zweite Zahl nicht angegeben wurde,

4. kann wie eine Stringvariable behandelt werden,

5. wird häufig verwendet, um eine Stringvariable Schritt für Schritt in einzelne Zeichen zu zerlegen.

Strings verbinden

Zum Anhängen einer Teilstringvariablen an eine andere.

1. Das **ATARI-BASIC** kennt hierfür keinen eigenen Befehl, sondern weist der ersten Stringvariablen hinter deren Ende die zweite Stringvariable zu.

2. Die verwendeten Teilstringvariablen müssen dimensioniert worden sein.

3. Es ist zulässig, mehrfach Teilstrings an dieselbe String-Variable anzuhängen.

   (siehe Befehl **DIM**)
   (Vorsicht bei der Zuweisung von Teilstrings die ein Vielfaches der Länge von 128 haben. Hier können Fehler auftreten.)

Strings vergleichen

Zum Vergleichen von Stringvariablen oder Textkonstanten.

1. Das **ATARI-BASIC** kennt hierfür keinen eigenen Befehl, sodern vergleicht Stringvariablen oder Texte mit Hilfe von logischen Operatoren Zeichen für Zeichen.

2. Dazu wird der Vergleich bei den jeweils ersten Zeichen begonnen und die ATASCII-Codes der beiden Zeichen werden verglichen. Durch den Aufbau des ATASCII-Codes werden auf diese Weise lexikalische Sortierungen möglich; der "kleinere String" bzw. Text kommt dabei an die erste Stelle.

3. Wer "größer" bzw. "kleiner" ist als der Andere, wird bei dem ersten Zeichen untersucht, an dem sich die Strings unterscheiden; dabei ist der ATASCII-Code des Zeichens maßgebend. So sind:
   a. Ziffern und Zeichen "kleiner als" Buchstaben,
   b. große Buchstaben "kleiner als" Kleinbuchstaben,
   c. generell die Zeichen "kleiner als" andere, deren ATASCII-Codezahl die kleinere ist.

ERKLÄRUNG

TASTE SYSTEM RESET

Zum Abbruch des Programmes in diesem Augenblick ("Warmstart").

Die Taste

1. bricht die Programmausführung im Augenblick des Drückens ab,
2. löscht das Programm und die Variablen nicht,
3. schliesst die Datenkanäle und schaltet die Tongeneratoren ab,
4. setzt den Grafikmodus auf 0 und löscht damit den Bildschirm,
5. setzt den Computer auf seine Anfangswerte (Randeinstellung, Grossbuchstaben, nicht invertierte Zeichendarstellung, Farbe usw.),
6. gibt die Tastatur nach dem Warmstart wieder für den Benutzer frei ("READY"),
7. führt in der Betriebsart "Bildschirmdarstellung" zu **BASIC** zurück,
8. kann zum "Zurückholen eines ausgestiegenen Rechners" benutzt werden.

ERKLÄRUNG

Taste TAB

Zum Bedienen der Tabulator-Funktion.

1. Nach dem Einschalten oder dem Drücken der Taste [SYSTEM RESET] werden vom Computer Tabulatormarken auf die Spalten 7, 15, 23, 31, und 39 gesetzt.

2. Wird nur die Taste [TAB] gedrückt, so springt der Cursor nach der jeweils folgenden Tabulatormarke.

3. Wird bei gedrückter Taste [CTRL] die Taste [TAB] gedrückt, so wird die Tabulatormarke, auf der der Cursor steht, gelöscht.

4. Wird bei gedrückter Taste [SHIFT] die Taste [TAB] gedrückt, so wird an der Cursorposition eine Tabulatormarke gesetzt.

5. Zum Programmieren der Tabulator-Funktionen wird der Befehl PRINT"TASTE [ESC] TAB-FUNKTION" mit der entsprechenden Anzahl von Leertasten verwendet für
   a. das Springen zur folgenden Marke Symbol ▷
   b. für das Löschen einer Marke Symbol [←]
   c. für Setzen einer Marke [→]
   innerhalb der Anführungszeichen werden die angegebenen Grafikzeichen geschrieben. Die Taste [ESC] muß vor jeder Tab-Funktion erneut gedrückt werden.

6. Die Standardwerte sollten gelöscht werden, bevor andere Marken verwendet werden.

Tabulator

Eine Einrichtung, die es ermöglicht, aufeinanderfolgende Zeilen als Textspalten anzuordnen, ohne daß zwischen den Textteilen einer Zeile Leerzeichen ausgegeben werden müssen.

Dazu gibt es die folgenden Methoden:

Der Cursor wird innerhalb der Zeile auf den Beginn der nächsten Spalte ("Tabulatormarke") gesetzt durch

1. den Befehl **POSITION** oder

2. die Taste [TAB], die auch programmiert werden kann,

   und dann der folgende Teil der Zeile mit dem Befehl **PRINT** ausgegeben.
   (siehe Taste [TAB]), Befehl **POSITION**

3a. verändern des Wertes in der Speicherstelle 85, die die horizontale Cursorposition enthält,

3b. verändern des Wertes in der Speicherzelle 84, die die vertikale Cursorposition enthält.

ERKLÄRUNG

TRAP (T.)

Zur Verhinderung eines Programmabbruchs bei der Programmdurchführung wenn ein Fehler aufgetreten ist.

Der Befehl

1. verhindert den Programmabbruch im Fehlerfall,
2. leitet die weitere Programmdurchführung zu der **BASIC**-Zeile um, deren Zeilennummer hinter **TRAP** angegeben wird ("bedingter Sprung" zur Fehlerbehandlung),
3. muß vor der Stelle im Programm stehen, an der der Fehler auftreten kann ("Einschalten" der Fehlerbehandlung),
4. muß nach jedem Fehlerfall erneut durchlaufen werden, damit er wieder wirksam wird ("Wiedereinschalten" der Fehlerbehandlung),
5. wird durch **TRAP** mit einer Zeilenzahl zwischen 32767 und 65535 unwirksam ("Ausschalten der Fehlerbehandlung"),
6. kann auch direkt eingegeben werden,
7. die Fehlerkennzahl eines aufgetretenen Fehlers findet man mit: PRINT PEEK(195).
8. Die Zeilennummer, in der der Fehler auftrat, findet man mit: PRINT 256 * PEEK(187) + PEEK(186).

ERKLÄRUNG

USR

Zum Aufruf eines in Maschinensprache geschriebenen Unterprogrammes durch ein **BASIC**-Programm.

Der Befehl

1. lässt den Rechner ein Maschinenprogramm ab der Startadresse ausführen (erster Wert in der Klammer hinter **USR**),

2. gestattet die Übergabe von mehreren Zahlen zwischen 0 und 65535 an das Maschinenprogramm (zweiter und folgende Werte in der Klammer hinter **USR**, durch Kommata getrennt),

3. gestattet die Rückgabe eines Ergebnisses an das **BASIC**-Programm über die numerische Variable vor **USR** (oder Speicherzellen 212 und 213),

4. gibt an das aufrufende **BASIC**-Programm zurück, sobald das Maschinenprogramm mit **RTS** beendet wird,

5. kann auch direkt eingegeben werden.

Ist der Beginn des Maschinenprogrammes im Speicher bekannt, wird der Dezimalwert der Speicherstelle als Adresse angegeben. Ist das Maschinenprogramm als STRING abgelegt, wird als Beginn die Adresse des STRINGS angegeben (siehe Befehl **ADR**).

ERKLÄRUNG



Zum Umwandeln einer String-Variablen in eine Zahl.

Die Funktion

1. macht eine Zahl aus dem ersten Ziffernteil der String-Variablen, die in der Klammer hinter **VAL** steht, wenn
   a. das erste Zeichen eine Ziffer ist oder
   b. die String-Variable mit einer Fliesskommazahl beginnt,

2. ergibt die Fehlermeldung 18, wenn das erste Zeichen keine Ziffer ist,

3. wird auch verwendet, um mit in String-Variablen enthaltenen Zahlen rechnen zu können.

   (siehe Befehl **STR$**)

ERKLÄRUNG

TASTE VIDEOUMKEHR

Zur Vertauschung ("Invertierung") der hellen und dunklen Bildpunkte der dargestellten Zeichen auf dem Bildschirm im Grafikmodus 0.

1. Beim Einschalten des Computers werden die Zeichen normal (hell auf dunklem Grund) dargestellt.

2. Wird bei ausgeschalteter Videoumkehr die Taste mit dem **ATARI**-Zeichen gedrückt, werden weitere Zeichen invertiert (dunkel auf hellem Grund, Videoumkehr) dargestellt.

3. Wird bei eingeschalteter Videoumkehr die Taste mit dem **ATARI**-Zeichen gedrückt, werden weitere Zeichen normal dargestellt.

4. Videoumkehr kann mit dem **PRINT**-Befehl programmiert werden.

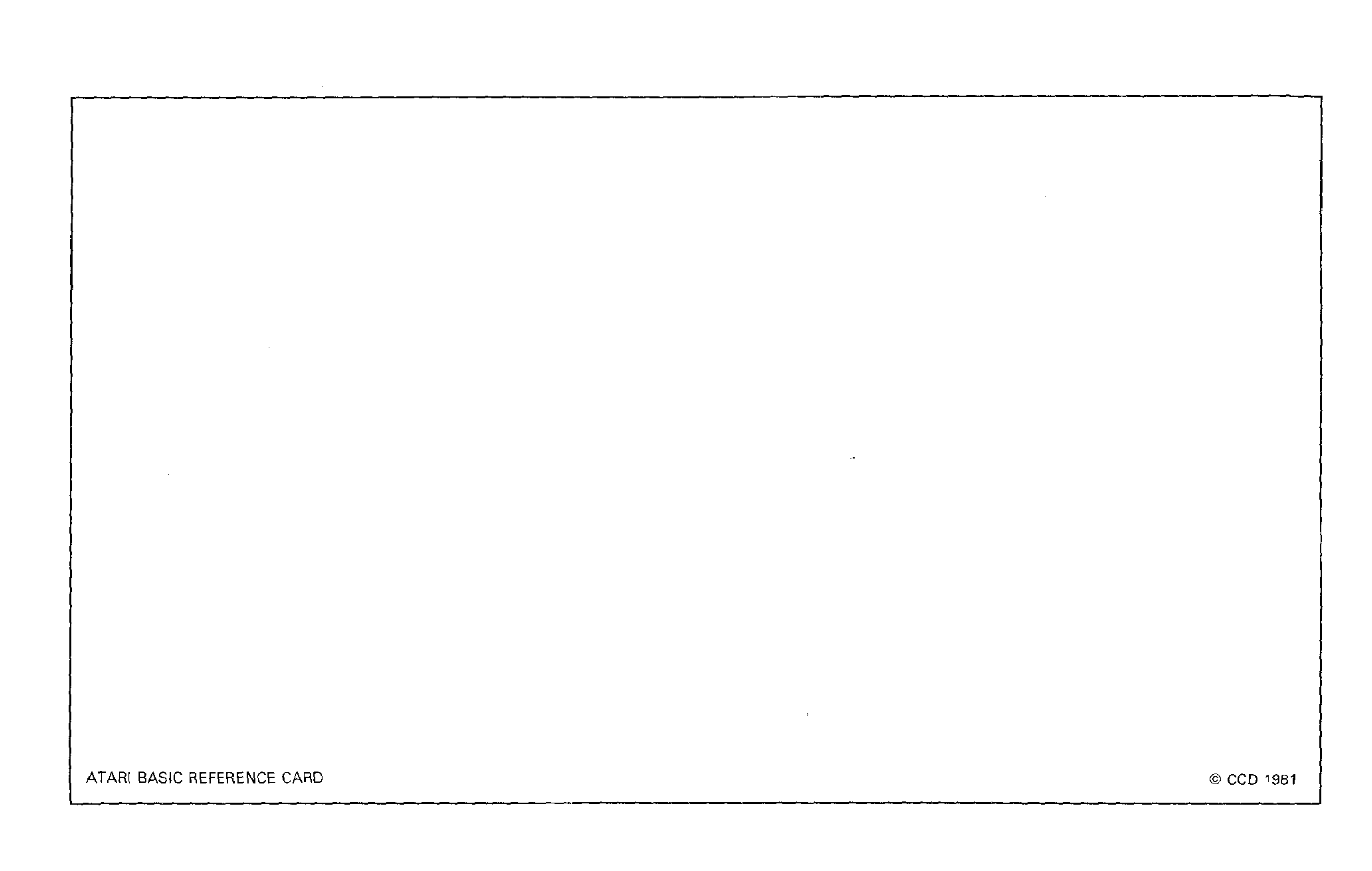
5. Die Darstellung der invertiert eigegebenen Zeichen kann auf verschiedene Weise dargestellt werden:
   a) POKE 755,0 stellt alle Zeichen nicht invertiert dar.
   b) POKE 755,1 stellt alle Zeichen als dunklen Zwischenraum dar.
   c) POKE 755,2 stellt alle Zeichen invertiert dar (wie nach dem Einschalten des Computers).
   d) POKE 755,3 stellt alle Zeichen als hellen Zwischenraum dar.
   e) addiert man zu den zu speichernden Werten der Punkte a bis d jeweils 4, so stehen alle Zeichen auf dem Kopf.

6. **BASIC**-Befehlsworte dürfen nur normal eingegeben werden.

Zur Ein- und Ausgabe bei Sonderaufgaben und im Grafikbereich.

Der Befehl

1. erlaubt z.B. bei der Benutzung von **DOS** die Verwendung als:
   a. OPEN,
   b. CLOSE,
   c. PUT,
   d. GET,
   e. Menü Funktion,
   f. POINT,
   g. NOTE,
   h. Statusabfrage (über die Fehlermeldung),
2. erlaubt bei der Benutzung der Farbgrafik die Verwendung als:
   a. Draw Line (siehe Befehl **DRAWTO**),
   b. FILL (siehe Befehl XIO18; dieser Befehl ist nur in der XIO-Form möglich),
3. verbindet die Kennummer der Sonderaufgabe mit der Nummer des Datenkanals und dem Namen der infrage kommenden Datei.
4. verarbeitet die Kennummer der Sonderaufgabe auch als Variable,
5. verarbeitet die Dateiangabe auch als String-Variable.


ATARI BASIC REFERENCE CARD
© CCD 1981

POP

Zur Rückkehr aus einem **BASIC**-Unterprogramm zu dem aufrufenden Programm, ohne den Befehl **RETURN** auszuführen.

Wenn aus einem Unterprogramm mit **GOSUB** zu einem **BASIC**-Unterprogramm gesprungen wird, wird in dem sogenannten "Stack-Speicher" die Zeilennummer gespeichert, an der das Hauptprogramm verlassen wurde. Am Ende des Unterprogramms bewirkt der Befehl **RETURN** die Rückkehr zum Hauptprogramm und dessen Fortsetzung bei der gespeicherten Zeilennummer (Normalfall).
Soll nun von einer beliebigen Stelle des Unterprogrammes zum Hauptprogramm zurückgesprungen werden (jedoch nicht zu der Zeile, an der das Hauptprogramm verlassen wurde), muß der Sprung mit den Befehlen **POP** und **GOTO** ausgeführt werden.

Die Befehle **POP** und **RETURN** entfernen jeweils die gespeicherte Zeilennummer aus dem Stack-Speicher.

1. Für jedes **GOSUB**, das nicht über **RETURN** zum Hauptprogramm zurückführt, muß ein **POP**-Befehl gegeben werden.

2. Der **POP**-Befehl muß sich im Zuge des weiteren Programmablaufes hinter **GOSUB** befinden, um wirksam zu werden.

ERKLÄRUNG

POSITION (POS.)

Zum Setzen des unsichtbaren Grafik-Cursors oder des Text-Cursors auf eine bestimmte Stelle des Bildschirms.

Der Befehl

1. wirkt in allen Grafikmodi (0-11) und Textmode,
2. stellt den unsichtbaren Grafik-Cursor oder den Text-Cursor an die hinter **POSITION** angegebene Stelle auf den Bildschirm,
3. enthält die horizontale Position an der ersten Stelle hinter **POSITION** [X] , Y,
4. enthält die vertikale Position an der zweiten Stelle hinter **POSITION** X , [Y],
5. wird in der Regel von dem Befehl **PRINT** gefolgt,
6. bewegt den Cursor erst, wenn ein Ein- oder Ausgabebefehl, der den Bildschirm betrifft, ausgeführt wird,
7. verwendet als Nullpunkt für die Positionsangaben die linke, obere Bildschirmecke (X=0 , Y=0).

ERKLÄRUNG

PRINT (PR oder ?)

Zur Ausgabe von Informationen auf den Bildschirm oder zu anderen Ausgabegeräten (siehe Befehl **LPRINT**).

Es können ausgegeben werden:

1. Texte aus Buchstaben und Zeichen im Klartext (z.B. "Otto"),
2. Texte aus Buchstaben und Zeichen als Stringvariablen (z.B. A$),
3. Zahlenwerte aus Variablen (z.B. A),
4. Zahlenwerte als Ergebnis eines Rechenvorganges (z.B. A+B),
5. Leerzeilen (nur PRINT angeben),
6. Ausgaben an andere Ausgabegeräte als den Bildschirm (siehe **OPEN/CLOSE**),
7. Cursorbewegungen,
8. Bildschirmlöschung,
9. Sonderzeichen (Summer, EOF etc., siehe ATASCII-Tabelle).

Jedes PRINT leitet eine neue Zeile ein. Soll in derselben Zeile weitergemacht werden, wird mit

10. ; an der folgenden Stelle derselben Zeile weitergedruckt,
11. , an der nächsten Tabulator-Position (siehe Taste [TAB]) weitergedruckt.

Pseudo-Grafik

Der Teil des Zeichensatzes, dessen ATASCII-Codezahlen zwischen 0 und 31 ("Steuerzeichen") liegen, bzw. 96 oder 123 sind. Er enthält Zeichen wie z.B. Balken verschiedener Dicke für waagerechte und senkrechte Verwendung, die Spielkartensymbole Kreuz, Pik, Herz und Karo usw.

Die Zeichen können

1. über den Befehl **PRINT** ausgegeben werden,
2. über Buchstaben- bzw. Punkt-, Komma-, oder Semikolon.Tasten eingegeben werden, wenn gleichzeitig die Taste [CTRL] gedrückt wird (siehe auch hinterer Deckel des Handbuches),

Die Tasten können

3. auf Dauer umgeschaltet werden, wenn gleichzeitig die Tasten [CTRL] und [CAPS/LOWR] gedrückt werden,
4. wieder auf kleine Buchstaben zurückgeschaltet werden, wenn die Taste [CAPS/LOWR] gedrückt wird,
5. wieder auf große Buchstaben zurückgeschaltet werden, wenn gleichzeitig die Tasten [SHIFT] und [CAPS/LOWR] gedrückt werden,
6. verwendet wird die Pseudografik u.a. für die einfache Herstellung von Textumrandungen.

ERKLÄRUNG

PTRIG

Zum Feststellen, ob der Druckknopf eines Drehreglers gedrückt ist.

Der Befehl

1. erlaubt die Auswahl eines der acht möglichen Drehregler, die jeweils zu zweit (von links nach rechts) an die vier Buchsen an der Vorderseite des Computers angeschlossen werden,

2. erwartet dazu
   a. eine Zahl,
   b. eine Variable oder
   c. einen zu berechnenden Wert
   zwischen 0 und 7 (von links nach rechts) in der Klammer hinter **PTRIG**,

3a. findet die Zahl 0, wenn der Knopf gedrückt ist,

3b. findet die Zahl 1, wenn der Knopf nicht gedrückt ist.

ERKLÄRUNG

PUT (PU.)

Zur Ausgabe eines Bytes über einen eröffneten Datenkanal.

Der Befehl

1. setzt einen mit **OPEN** eröffneten Datenkanal voraus (Kennzahl 8, 12 oder 9),

2. muß mit # und der Nummer des Datenkanals versehen werden, über den ausgegeben werden soll,

3. gibt ein Byte aus (8-Bit-Parallelwort, entspricht einer Zahl zwischen 0 und 255), das
   a) eine Zahl,
   b) eine Variable
   c) oder ein zu berechnender Wert sein kann.

(siehe Befehl **GET**)

ERKLÄRUNG

RAD

Zum Umschalten der Winkelfunktionen auf das Bogenmaß.

Der Befehl

1. schaltet die Winkelfunktionen auf das Bogenmaß (Kreisumfang entspricht zweimal π),

2. wird nicht aufgehoben, wenn die Taste [SYSTEM RESET] gedrückt wird,

3. ist eingeschaltet, wenn der Coputer eingeschaltet wird.

ERKLÄRUNG

READ (REA.)/DATA (D.)

Zur Zuweisung von Werten, die aus einer **DATA**-Liste gelesen werden, zu Variablen.

Der Befehl

1. liest nacheinander die durch Kommas getrennten Werte oder Zeichen aus den **DATA**-Zeilen in eine oder mehrere, durch Kommas getrennte Variablen, die hinter **READ** angegeben werden,

2. stellt bei jedem Lesen eines Wertes oder Zeichens den **READ/DATA**-Zeiger um eins weiter (siehe Befehl **RESTORE**),

3. erwartet, daß nicht mehr Werte durch **READ** gelesen werden sollen, als in der **DATA**-Liste noch vorhanden sind (Listenende markieren),

4. erwartet, daß die Variablenart hinter **READ** den zu lesenden Werten entspricht,

5. wird in der Regel programmiert (dabei kann die **DATA**-Liste irgendwo im Programm stehen),

6. kann auch direkt eingegeben werden (dabei muß eine zu lesende **DATA**-Zeile bereits vorher im Programm ausgeführt worden sein),

7. läßt hinter **DATA** als zu lesende Daten keine Kommas zu; Anführungszeichen werden ausgegeben.

REM (R. oder .LEERTASTE)

Zur Darstellung von Erläuterungen ("REMARKS") im Programm.

Der Befehl

1. erlaubt die Darstellung von Erläuterungen in normalen **BASIC**-Zeilen,

2. verhindert die Bearbeitung des Teiles der **BASIC**-Zeile, der hinter **REM** steht,

3. kann auch dazu verwendet werden, beim Austesten eine **BASIC**-Zeile durch davorsetzen von **REM** zu sperren, ohne sie zu löschen,

4. wird jedoch beim Auslisten mit ausgegeben,

5. kann nur programmiert werden.

ERKLÄRUNG

RESTORE (RES.)

Zum Setzen des READ/DATA-Zeigers auf die DATA-Zeile, mit der der Lesevorgang beginnen soll (siehe Befehl **READ/DATA**).

Der Befehl

1. setzt den READ/DATA-Zeiger auf den ersten Wert in der ersten DATA-Zeile des Programms, wenn nur **RESTORE** allein angegeben wird,

2. setzt den READ/DATA-Zeiger auf den ersten Wert der DATA-Zeile, deren Zeilennummer hinter **RESTORE** angegeben wird,

3. erlaubt den wiederholten Gebrauch derselben DATA-Liste,

4. erlaubt das gezielte, schnelle Auffinden von Daten in einer DATA-Liste, wenn die Zeilennummer bekannt ist, in der der Wert steht.

ERKLÄRUNG

Taste RETURN

Zum Beenden einer Zeile bei der Eingabe.

Die Taste

1. beendet die Eingabe einer BASIC-Zeile bei der Programmeingabe; ihre Funktion ist hier mit **POKE** 764,12 programmierbar; danach erscheint **"READY"**, die Ausführung erscheint eine Zeile unter der Meldung **READY**,

2. beendet eine Zeile bei der Direkteingabe; danach erscheint **"READY"** nicht,

3. beendet eine Eingabe während der Programmdurchführung als Antwort auf den Befehl **INPUT**; danach wird mit der Programmdurchführung fortgefahren,

4. hat im ATASCII-Code die Bezeichnung **EOL** ("END-OF-LINE") und die Codezahl 155,

5. bringt den Cursor nach Abschluß der Zeile zum Beginn der folgenden Zeile.

(siehe Betriebsarten und BASIC Zeilen-Format)

ERKLÄRUNG

Zum Erzeugen einer Zufallszahl.

Die Funktion

1. erzeugt eine Zufallszahl zwischen 0 und 1,
2. erzeugt nie eine 1,
3. benötigt in der Klammer hinter **RND** immer eine Dummy-Variable oder -Zahl, die aber ohne Einfluß bleibt,
4. wird häufig in Spielprogrammen in Verbindung mit anderen **BASIC**-Statements als Würfel verwendet.

ERKLÄRUNG

RUN (RU.)

Der Befehl

1. beinhaltet **CLR** und **RESTORE**,

2. setzt den Stack zurück (siehe Befehle **FOR/NEXT** und **GOSUB**),

3. schliesst alle offenen Datenkanäle und schaltet alle Tongeneratoren aus,

4. beginnt die Programmdurchführung eines gespeicherten Programmes, wenn nach **RUN** keine Dateiangaben gemacht worden sind oder

5. beginnt die Programmdurchführung, nachdem das Programm von dem hinter **RUN** in den Dateiangaben bezeichneten Datei geladen worden ist, wenn das betreffende Programm mit **SAVE** gespeichert worden war,

6. kann das Programm nur bei der niedrigsten Zeilennummer starten (siehe Befehl **GOTO**),

7. kann programmiert werden.

Die Programmdurchführung läuft bis zu einem **STOP**- oder **END**-Befehl, bis zu einem Fehler (siehe Befehl **TRAP**) oder bis die Tasten [BREAK] oder [SYSTEM RESET] gedrückt wurden.

ERKLÄRUNG

SAVE (S.)

Zum Schreiben eines im Speicher stehenden Programmes in übersetzter Form zum Kassettenrekorder oder zur Diskettenstation.

Der Befehl

1. verändert ein im Speicher stehendes Programm samt Variablen nicht,
2. schreibt das Programm in übersetzter Form (Tokenformat) (siehe Befehle **CSAVE** und **LIST**),
3. gibt an die Kassette aus, wenn "C : " hinter **SAVE** angegeben wurde,
4. gibt an die Diskette aus, wenn "D : " hinter **SAVE** angegeben wurde,
5. muß für die Diskette mit einem Programmnamen versehen werden,
6. wird in der Regel direkt eingegeben,
7. kann auch programmiert werden,
8. schreibt langsamer als **CSAVE**, weil die Pausen zwischen den Daten-Blöcken länger sind.

ERKLÄRUNG

SETCOLOR (SE.)

Zum Festlegen der Farbe und Helligkeit, die in einem bestimmten Farbregister ("Farbeimer") gespeichert werden soll.

Der Befehl

1. füllt die Farbregister ("Farbeimer") mit den gewünschten Farb- und Helligkeitswerten (bereits dargestellte farbige Bildteile verändern ihre Farbe durch nachträgliches Ändern der Werte im Farbregister),
2. wirkt in allen Grafikmodi,
3. enthält die Kennzahlen für
   a. das Farbregister (0 bis 4) an erster Stelle
   b. die Farbe (0 bis 15) an zweiter Stelle
   c. die Helligkeit (gerade Zahl von 0=dunkel bis 14=hell) an dritter Stelle
   hinter **SETCOLOR** a, b, c,
4. die folgenden Anfangswerte werden vom Computer in die Farbregister geschrieben

a.	SETCOLOR 0 , 2 , 8	(orange),
b.	SETCOLOR 1 , 12 , 10	(hellgrün),
c.	SETCOLOR 2 , 9 , 4	(dunkelblau), Hintergrundfarbe im Grafikmode 0 und für alle Textfenster
d.	SETCOLOR 3 , 4 , 6	(rot),
e.	SETCOLOR 4 , 0 , 0	(schwarz). Randfarbe im Grafikmode 0 bzw. Hintergrundfarbe des Grafikfensters

(Diese Farben werden automatisch nach dem Einschalten des Computers und nach Betätigung der Taste [SYSTEM RESET] gesetzt. Siehe auch Befehle **GRAPHICS 0-11** und **COLOR**.)

ERKLÄRUNG

SGN

Zum Feststellen des Vorzeichens einer Zahl.

Die Funktion

1. stellt das Vorzeichen fest von
   a. einer Zahl
   b. einer Variablen oder
   c. einem zu berechnenden Wert,

2. ergibt −1 , wenn der Wert in Klammern kleiner als 0 ist,

3. ergibt 0 , wenn der Wert in Klammern gleich 0 ist,

4. ergibt +1 , wenn der Wert in Klammern größer als 0 ist.

ERKLÄRUNG

Taste SHIFT

Zum Umschalten der Funktion einiger Tasten (erste Umschaltebene).

Wird die Taste [SHIFT] gleichzeitig gedrückt mit

1. einer Buchstabentaste, wenn auf Kleinbuchstaben geschaltet war, so wird der entsprechende Großbuchstabe dargestellt,

2. der Taste [TAB], so wird an die Cursorfunktion eine Tabulatormarke gesetzt,

3. der Taste [CLEAR], so wird der Bildschirm gelöscht,

4. der Taste [INSERT], so wird an der Cursorposition eine Leerzeile eingeschoben; der restliche Bildschirminhalt wird nach unten verschoben; dabei geht die Zeile am unteren Rand verloren,

5. der Taste [DELETE BACK'S], so wird die Zeile an der Cursorposition entfernt; der Rest des Bildschirminhalts rückt eine Zeile nach oben.

ERKLÄRUNG **SIN**

Zum Errechnen des Sinus.

Die Funktion

1. errechnet den Sinus
   a. der Zahl,
   der Variablen oder
   c. des zu berechnenden Wertes,

die,der in der Klammer hinter **SIN** steht,

2. ergibt immer Werte zwischen −1 und +1,

3. kann den Winkel im Bogenmaß oder in Grad verarbeiten.

(siehe Befehle **DEG** und **RAD**)

ERKLÄRUNG

SOUND (SO.)

Zum Erzeugen von Tönen, Klängen und Geräuschen.

Der Befehl

1. erlaubt die Anwahl eines der vier vorhandenen Tongeneratoren durch die erste Zahl (0 bis 3) hinter **SOUND**,
2. erlaubt die Steuerung der Tonhöhe durch die zweite Zahl (0≙hoch bis 255≙tief) hinter **SOUND**,
3. erlaubt die Steuerung des Verzerrungsgrades des Tones durch die dritte Zahl (0 bis 14) hinter **SOUND**,
4. erlaubt die Steuerung der Lautstärke durch die vierte Zahl (1≙leise bis 15≙laut) hinter **SOUND**,
5. gestattet das Ausschalten des angewählten Tongenerators durch einen neuen **SOUND**-Befehl mit der Lautstärke 0; sollen alle Tongeneratoren gleichzeitig ausgeschaltet werden, wird der Befehl **END** benutzt,
6. erlaubt die gleichzeitige Benutzung aller vier Tongeneratoren.

Es können die genannten Zahlen an allen vier Stellen durch Variablen oder zu berechnende Werte ersetzt werden. Die Töne können über den Lautsprecher des FS-Gerätes gehört, und beim ATARI 800 zusätzlich an der Monitorbuchse abgenommen werden.

SOUND (SO.), Erg. 1

Mit den vier Tongeneratoren können musikalische Noten gespielt werden, wenn für die zweite Zahl hinter **SOUND** die entsprechenden Werte eingesetzt werden. **POKE 53768,1** bzw. **POKE 53768,4** verschieben den Tonbereich nach unten bzw. oben.

Tonhöhe	Zahl	Tohnhöhe	Zahl	Tonhöhe	Zahl
tiefes C	243	kleines C	121	hohes c′	60
Cis	230	cis	114	cis′	57
D	217	d	108	d′	53
Dis	204	dis	102	dis′	50
E	193	e	96	e′	47
F	182	f	91	f′	45
Fis	173	fis	85	fis′	42
G	162	g	81	g′	40
Gis	153	gis	76	gis′	37
A	144	a	72	a′	35
Ais	136	ais	68	ais′	33
H	128	h	64	h′	31
C	121	c′	60	c″	29

ERKLÄRUNG

SQR

Zum Errechnen der Quadratwurzel einer Zahl.

Die Funktion

1. errechnet die Quadratwurzel der Zahl, die in der Klammer hinter **SQR** steht,
2. setzt voraus, daß der Wert in Klammern positiv ist.

ERKLÄRUNG **STACK**

Speicherbereich im Computer, in dem von der CPU u. a. beim Einsprung in Unterprogramme die Rücksprungadressen abgelegt werden.

Der Stack

1. liegt beim ATARI 400/800 im Speicherbereich zwischen 100 und 1FF (Hexadezimal) bzw. 256 und 511 (dezimal),

2. wird von der CPU mit Hilfe eines eigenen Zählers (des Stackpointers) beschrieben und gelesen,

3. wird von der höchsten Speicherstelle nach tieferen fortschreitend beschrieben und gelesen,

4. arbeitet nach dem Prinzip "als Erster hinein, als Letzter hinaus",

5. übernimmt beim Einsprung in ein Unterprogramm die Rücksprungadresse, damit nach abgearbeitetem Unterprogramm das Hauptprogramm an der richtigen Stelle fortgesetzt werden kann,

6. enthält noch die alte Rücksprungadresse, wenn ein Unterprogramm nicht mit dem vorgesehenem Befehl (**RTS** bzw. **RETURN**) verlassen wurde, was bei geschachtelten Unterprogrammen beim nächsten **RTS**- bzw. **RETURN**-Befehl mit einem Rücksprung zur falschen Stelle endet (siehe Befehl **POP**),

7. kann zur Übergabe von Parametern zwischen Unterprogrammen benutzt werden (siehe Befehl **USR**).

ERKLÄRUNG

STATUS (ST.)

Zum Feststellen des Betriebszustandes eines Datenkanals (z.B. mit dem Kennzeichen "D : ", "P : " oder "R : ").

Der Befehl

1. setzt einen eröffneten Datenkanal voraus,

2. verwendet die Nummer des Datenkanals hinter #,

3. liest die Kennzahl des in dem genannten Datenkanal aufgetretenen Fehlers in die Variable,

4. erkennt Fehler im Zusammenhang mit
   a. der Diskettenstation,
   b. dem Drucker,
   c. dem Schnittstellenumsetzer.

(siehe **ERROR)**

STICK

Zum Feststellen der Stellung eines Steuerknüppels.

Der Befehl

1. erlaubt die Auswahl eines der vier möglichen Steuerknüppel, die von links nach rechts an die vier Buchsen an der Vorderseite des Computers angeschlossen werden,

2. erwartet dazu
   a. eine Zahl,
   b. eine Variable oder
   c. einen zu berechnenden Wert
   zwischen 0 und 3 (von links nach rechts) in der Klammer hinter **STICK,**

3. findet die angegebene Zahl, wenn der Steuerknüppel in die angegebene Richtung gedrückt wird:

15	in Ruhe
14	nach vorn
13	zurück
7	nach rechts
11	nach links
6	rechts vorn
10	links vorn
9	links zurück
5	rechts zurück

ERKLÄRUNG

STOP (STO.)

Zur Unterbrechung der Programmausführung an dieser Stelle.

Der Befehl

1. unterbricht die Programmausführung an dieser Stelle,
2. löscht das Programm und die Variablen nicht,
3. läßt Datenkanäle, Tongeneratoren und Grafikmodus unverändert,
4. wird im Programm beim Austesten verwendet,
5. kann nur programmiert werden (siehe Taste [BREAK]),
6. kann wieder rückgängig gemacht werden (siehe Befehl **CONT**),
7. gibt nach der Programmunterbrechung die Zeile an, in der das Programm angehalten hat,
8. gibt die Tastatur wieder für den Benutzer zur Programmierung frei.

ERKLÄRUNG

STR$

Zum Umwandeln einer Zahl in eine String-Variable.

Die Funktion

1. macht eine String-Variable aus der Zahl, die in der Klammer hinter **STR$** steht,
2. kann wie eine String-Variable behandelt werden,
3. darf in einer Vergleichsoperation nur einmal vorkommen.

   (siehe Befehl **VAL**)

ERKLÄRUNG

STRIG

Zum Feststellen, ob der Feuerknopf eines Steuerknüppels gedrückt ist.

Der Befehl

1. erlaubt die Auswahl eines der vier möglichen Steuerknüppel, die von links nach rechts an die vier Buchsen an der Vorderseite des Computers angeschlossen werden,

2. erwartet dazu
   a. eine Zahl,
   b. eine Variable oder
   c. einen zu berechnenden Wert
   zwischen 0 und 3 (von links nach rechts) in der Klammer hinter **STRIG,**

3. findet die Zahl
   a. 0, wenn der Feuerknopf gedrückt ist,
   b. 1, wenn der Feuerknopf nicht gedrückt ist.

ERKLÄRUNG

STRING-VARIABLEN

Variablen sind die Namen des Inhaltes von Speicherplätzen. Der Inhalt eines solchen Speicherplatzes kann mit Hilfe des Variablennamens gelesen, beschrieben oder verändert werden.

Variablennamen dürfen bis zu 120 Zeichen lang sein, müssen mit einem Buchstaben beginnen und dürfen nur Grossbuchstaben und Ziffern enthalten. Es sollen keine **BASIC**-Befehlsworte als erster Teil des Namens verwendet werden.

Es können insgesamt bis zu 128 Variablennamen aller Variablenarten zugewiesen werden.

1. Eine STRING-VARIABLE enthält Text (z.B. "Otto"), also Buchstaben, Ziffern oder Sonderzeichen ("STRING"). Ihr Name endet immer mit $, sie muss "dimensioniert" werden (siehe Befehl **DIM**).

2. In einer **STRING-VARIABLEN** können nur soviele Zeichen abgelegt werden, wie mit dem **DIM**-Befehl vorgegeben wurde.

3. Der Variablenname kann bei der Ausgabe mit einem **PRINT**-Befehl wie ein Text behandelt werden.

4. Es können Untermengen angesprochen und String-Verknüpfungen durchgeführt werden

Strings teilen

Zum Herstellen einer Teilstringvariablen aus einer Stringvariablen.

Der Befehl

1. erzeugt eine Teilstringvariable aus der angegebenen Stringvariablen,

2. beginnt den Teilstring an der Stelle des Strings, die durch die Zahl in Klammern hinter dem Stringnahmen angegeben wird,

3. beendet den Teilstring
   a. an der Stelle des Strings, die durch die zweite Zahl in der Klammer hinter dem Stringnahmen angegeben wurde oder
   b. am Ende des Strings, wenn in der Klammer eine zweite Zahl nicht angegeben wurde,

4. kann wie eine Stringvariable behandelt werden,

5. wird häufig verwendet, um eine Stringvariable Schritt für Schritt in einzelne Zeichen zu zerlegen.

Strings verbinden

Zum Anhängen einer Teilstringvariablen an eine andere.

1. Das **ATARI-BASIC** kennt hierfür keinen eigenen Befehl, sondern weist der ersten Stringvariablen hinter deren Ende die zweite Stringvariable zu.

2. Die verwendeten Teilstringvariablen müssen dimensioniert worden sein.

3. Es ist zulässig, mehrfach Teilstrings an dieselbe String-Variable anzuhängen.

   (siehe Befehl **DIM**)
   (Vorsicht bei der Zuweisung von Teilstrings die ein Vielfaches der Länge von 128 haben. Hier können Fehler auftreten.)

Strings vergleichen

Zum Vergleichen von Stringvariablen oder Textkonstanten.

1. Das **ATARI-BASIC** kennt hierfür keinen eigenen Befehl, sodern vergleicht Stringvariablen oder Texte mit Hilfe von logischen Operatoren Zeichen für Zeichen.

2. Dazu wird der Vergleich bei den jeweils ersten Zeichen begonnen und die ATASCII-Codes der beiden Zeichen werden verglichen. Durch den Aufbau des ATASCII-Codes werden auf diese Weise lexikalische Sortierungen möglich; der "kleinere String" bzw. Text kommt dabei an die erste Stelle.

3. Wer "größer" bzw. "kleiner" ist als der Andere, wird bei dem ersten Zeichen untersucht, an dem sich die Strings unterscheiden; dabei ist der ATASCII-Code des Zeichens maßgebend. So sind:
   a. Ziffern und Zeichen "kleiner als" Buchstaben,
   b. große Buchstaben "kleiner als" Kleinbuchstaben,
   c. generell die Zeichen "kleiner als" andere, deren ATASCII-Codezahl die kleinere ist.

ERKLÄRUNG



Zum Abbruch des Programmes in diesem Augenblick ("Warmstart").

Die Taste

1. bricht die Programmausführung im Augenblick des Drückens ab,
2. löscht das Programm und die Variablen nicht,
3. schliesst die Datenkanäle und schaltet die Tongeneratoren ab,
4. setzt den Grafikmodus auf 0 und löscht damit den Bildschirm,
5. setzt den Computer auf seine Anfangswerte (Randeinstellung, Grossbuchstaben, nicht invertierte Zeichendarstellung, Farbe usw.),
6. gibt die Tastatur nach dem Warmstart wieder für den Benutzer frei ("READY"),
7. führt in der Betriebsart "Bildschirmdarstellung" zu **BASIC** zurück,
8. kann zum "Zurückholen eines ausgestiegenen Rechners" benutzt werden.

ERKLÄRUNG

Taste TAB

Zum Bedienen der Tabulator-Funktion.

1. Nach dem Einschalten oder dem Drücken der Taste [SYSTEM RESET] werden vom Computer Tabulatormarken auf die Spalten 7, 15, 23, 31, und 39 gesetzt.

2. Wird nur die Taste [TAB] gedrückt, so springt der Cursor nach der jeweils folgenden Tabulatormarke.

3. Wird bei gedrückter Taste [CTRL] die Taste [TAB] gedrückt, so wird die Tabulatormarke, auf der der Cursor steht, gelöscht.

4. Wird bei gedrückter Taste [SHIFT] die Taste [TAB] gedrückt, so wird an der Cursorposition eine Tabulatormarke gesetzt.

5. Zum Programmieren der Tabulator-Funktionen wird der Befehl PRINT"TASTE [ESC] TAB-FUNKTION" mit der entsprechenden Anzahl von Leertasten verwendet für
   a. das Springen zur folgenden Marke Symbol ▷
   b. für das Löschen einer Marke Symbol [←]
   c. für Setzen einer Marke [→]
   innerhalb der Anführungszeichen werden die angegebenen Grafikzeichen geschrieben. Die Taste [ESC] muß vor jeder Tab-Funktion erneut gedrückt werden.

6. Die Standardwerte sollten gelöscht werden, bevor andere Marken verwendet werden.

ERKLÄRUNG

Tabulator

Eine Einrichtung, die es ermöglicht, aufeinanderfolgende Zeilen als Textspalten anzuordnen, ohne daß zwischen den Textteilen einer Zeile Leerzeichen ausgegeben werden müssen.

Dazu gibt es die folgenden Methoden:

Der Cursor wird innerhalb der Zeile auf den Beginn der nächsten Spalte ("Tabulatormarke") gesetzt durch

1. den Befehl **POSITION** oder

2. die Taste [TAB], die auch programmiert werden kann,

   und dann der folgende Teil der Zeile mit dem Befehl **PRINT** ausgegeben.
   (siehe Taste [TAB]), Befehl **POSITION**

3a. verändern des Wertes in der Speicherstelle 85, die die horizontale Cursorposition enthält,

3b. verändern des Wertes in der Speicherzelle 84, die die vertikale Cursorposition enthält.

ERKLÄRUNG

TRAP (T.)

Zur Verhinderung eines Programmabbruchs bei der Programmdurchführung wenn ein Fehler aufgetreten ist.

Der Befehl

1. verhindert den Programmabbruch im Fehlerfall,
2. leitet die weitere Programmdurchführung zu der **BASIC**-Zeile um, deren Zeilennummer hinter **TRAP** angegeben wird ("bedingter Sprung" zur Fehlerbehandlung),
3. muß vor der Stelle im Programm stehen, an der der Fehler auftreten kann ("Einschalten" der Fehlerbehandlung),
4. muß nach jedem Fehlerfall erneut durchlaufen werden, damit er wieder wirksam wird ("Wiedereinschalten" der Fehlerbehandlung),
5. wird durch **TRAP** mit einer Zeilenzahl zwischen 32767 und 65535 unwirksam ("Ausschalten der Fehlerbehandlung"),
6. kann auch direkt eingegeben werden,
7. die Fehlerkennzahl eines aufgetretenen Fehlers findet man mit: PRINT PEEK(195).
8. Die Zeilennummer, in der der Fehler auftrat, findet man mit: PRINT 256 * PEEK(187) + PEEK(186).

ERKLÄRUNG

USR

Zum Aufruf eines in Maschinensprache geschriebenen Unterprogrammes durch ein **BASIC**-Programm.

Der Befehl

1. lässt den Rechner ein Maschinenprogramm ab der Startadresse ausführen (erster Wert in der Klammer hinter **USR**),

2. gestattet die Übergabe von mehreren Zahlen zwischen 0 und 65535 an das Maschinenprogramm (zweiter und folgende Werte in der Klammer hinter **USR**, durch Kommata getrennt),

3. gestattet die Rückgabe eines Ergebnisses an das **BASIC**-Programm über die numerische Variable vor **USR** (oder Speicherzellen 212 und 213),

4. gibt an das aufrufende **BASIC**-Programm zurück, sobald das Maschinenprogramm mit **RTS** beendet wird,

5. kann auch direkt eingegeben werden.

Ist der Beginn des Maschinenprogrammes im Speicher bekannt, wird der Dezimalwert der Speicherstelle als Adresse angegeben. Ist das Maschinenprogramm als STRING abgelegt, wird als Beginn die Adresse des STRINGS angegeben (siehe Befehl **ADR**).

ERKLÄRUNG

VAL

Zum Umwandeln einer String-Variablen in eine Zahl.

Die Funktion

1. macht eine Zahl aus dem ersten Ziffernteil der String-Variablen, die in der Klammer hinter **VAL** steht, wenn
   a. das erste Zeichen eine Ziffer ist oder
   b. die String-Variable mit einer Fliesskommazahl beginnt,

2. ergibt die Fehlermeldung 18, wenn das erste Zeichen keine Ziffer ist,

3. wird auch verwendet, um mit in String-Variablen enthaltenen Zahlen rechnen zu können.

   (siehe Befehl **STR$**)

ERKLÄRUNG

TASTE VIDEOUMKEHR

Zur Vertauschung ("Invertierung") der hellen und dunklen Bildpunkte der dargestellten Zeichen auf dem Bildschirm im Grafikmodus 0.

1. Beim Einschalten des Computers werden die Zeichen normal (hell auf dunklem Grund) dargestellt.
2. Wird bei ausgeschalteter Videoumkehr die Taste mit dem **ATARI**-Zeichen gedrückt, werden weitere Zeichen invertiert (dunkel auf hellem Grund, Videoumkehr) dargestellt.
3. Wird bei eingeschalteter Videoumkehr die Taste mit dem **ATARI**-Zeichen gedrückt, werden weitere Zeichen normal dargestellt.
4. Videoumkehr kann mit dem **PRINT**-Befehl programmiert werden.
5. Die Darstellung der invertiert eigegebenen Zeichen kann auf verschiedene Weise dargestellt werden:
   a) POKE 755,0 stellt alle Zeichen nicht invertiert dar.
   b) POKE 755,1 stellt alle Zeichen als dunklen Zwischenraum dar.
   c) POKE 755,2 stellt alle Zeichen invertiert dar (wie nach dem Einschalten des Computers).
   d) POKE 755,3 stellt alle Zeichen als hellen Zwischenraum dar.
   e) addiert man zu den zu speichernden Werten der Punkte a bis d jeweils 4, so stehen alle Zeichen auf dem Kopf.
6. **BASIC**-Befehlsworte dürfen nur normal eingegeben werden.

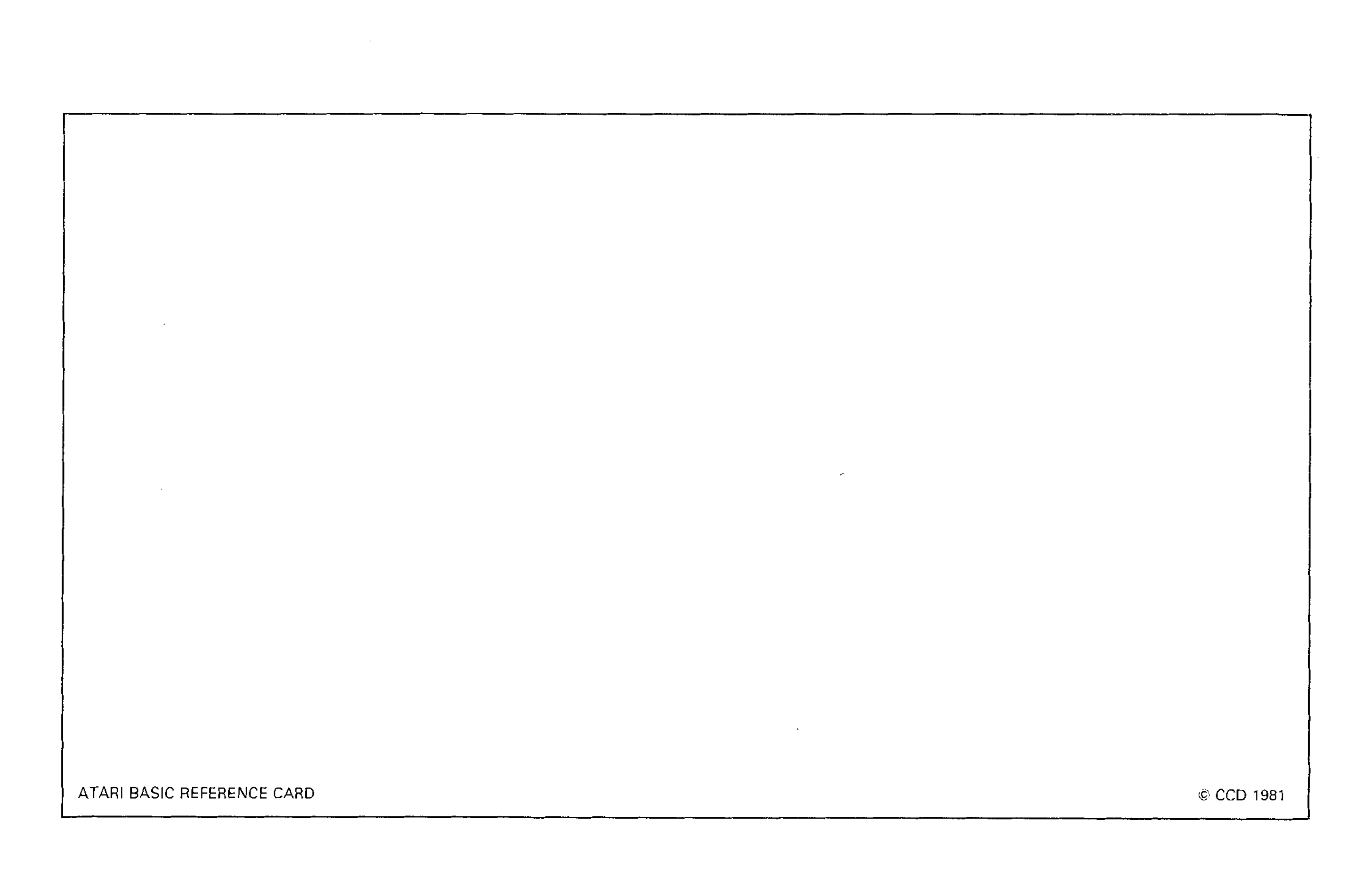
ERKLÄRUNG

XIO (X.)

Zur Ein- und Ausgabe bei Sonderaufgaben und im Grafikbereich.

Der Befehl

1. erlaubt z.B. bei der Benutzung von **DOS** die Verwendung als:
   a. OPEN,
   b. CLOSE,
   c. PUT,
   d. GET,
   e. Menü Funktion,
   f. POINT,
   g. NOTE,
   h. Statusabfrage (über die Fehlermeldung),
2. erlaubt bei der Benutzung der Farbgrafik die Verwendung als:
   a. Draw Line (siehe Befehl **DRAWTO**),
   b. FILL (siehe Befehl XIO18; dieser Befehl ist nur in der XIO-Form möglich),
3. verbindet die Kennummer der Sonderaufgabe mit der Nummer des Datenkanals und dem Namen der infrage kommenden Datei.
4. verarbeitet die Kennummer der Sonderaufgabe auch als Variable,
5. verarbeitet die Dateiangabe auch als String-Variable.



ATARI BASIC REFERENCE CARD